Statistical Techniques in Robotics (16-831, S22) Lecture #10 (Monday, February 21, 2022)

Support Vector Machines

Lecturer: Kris Kitani *Scribes: Michelle Zhao, Shane Deng*

1 Review

In the last lectures, we covered the algorithms of Gradient Descent, and Stochastic Gradient Descent (SGD). Gradient descent is an approach for minimizing convex, differentiable functions. Alternative gradient descent algorithms stochastically sample the data (SGD) or observe the data in mini-batches (Batch Gradient Descent), while maintaining convergence to the minimum solution. We observed how Stochastic Gradient Descent can be represented as an online convex optimization problem, and in particular, is a special case of Online Mirror Descent (OMD).

1.1 Gradient Descent

The intuition of gradient descent, is that if we want to locate the minimum of some objective function f, we must move in the direction opposite to the gradient. The gradient with respect to the entire dataset. Then, the weights are updated by subtracting the gradient from the current set of weights.

Algorithm 1 Gradient Descent

$\mathbf{w}^{(0)} \leftarrow 0 \in R^N$
$\eta > 0$
for $t = 0, \cdots, T-1$ **do**
  $\mathbf{v}^{(t)} = \nabla f(\mathbf{w}^{(t-1)})$
  $\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta \mathbf{v}^{(t)}$ {Mirror projection}
end for

1.2 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a convex optimizer that does not require the exact gradient, but optimizes by taking steps in the gradient direction. In each iteration of SGD, we sample from a data distribution, obtaining a single sample or minibatch of samples. Then, we compute the gradient with respect to the sampled point, which in expectation equals the true gradient. Then, the weights are updated by subtracting the gradient from the current set of weights, which optimizes in the direction opposite to the gradient.

Algorithm 2 Stochastic Gradient Descent

$\mathbf{w}^{(0)} \leftarrow 0 \in R^N$
$\eta > 0$
for $t = 0, \cdots, T-1$ **do**
  $z \sim \mathcal{D}$ {Sample from data distribution}
  $\mathbf{v}^{(t)} = \nabla f_z(\mathbf{w}^{(t-1)})$ {Expectation is true gradient}
  $\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \eta \mathbf{v}^{(t)}$ {Mirror projection}
end for

SGD achieves similar convergence bounds as gradient descent, which computes $\mathbf{v}^{(t)}$ over the entire dataset.

2 Online SVM

2.1 Convex Functions

Recall from the previous lecture:
A function $f : S \to R$ is a convex function if it satisfies:

$$f(\alpha * w + (1-\alpha) * v) \leq \alpha * f(w) + (1-\alpha) * f(v) \; for \; all \; \alpha \in [0,1]$$

A key property is that the convex function has a global minimum which means it is suitable for optimization problems.

2.2 Hyperplanes

A hyperplane is a subspace whose dimension is $n-1$ of the main space. For example, in a 2D space (e.g a 2D surface), the hyperplane is a 1D line. More importantly, a hyperplane is a subspace that divides the data points into n sections.

2.2.1 Hyperplanes in 2D

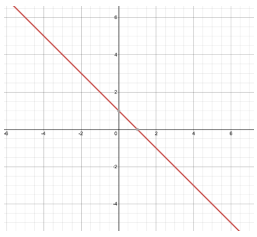

Figure 1: Hyperplane in 2D

A standard representation of a line is $w_1 x_1 + w_2 x_2 + b = 0$. A line can be written as a dot product plus a bias.

$$w \cdot x + b = 0 \text{ (offset / bias outside)}$$

$$w \in \mathbb{R}^2$$

Another version is the 3-dimensional version: add a weight 1 to the x vector, and push the bias inside of the weights.

$$w \cdot x = 0 \text{ (offset / bias inside)}$$

$$w \in \mathbb{R}^3$$

Note: An important property of this hyperplane is that you are free to choose any normalization you want on **w**. You can multiply this side of the equation by any scalar λ, and the equation still describes the same line.

$$w_1 x_1 + w_2 x_2 + b = 0$$

$$\lambda(w_1 x_1 + w_2 x_2 + b) = 0$$

$$w \cdot x + b = 0$$

- **Vertical distance between the origin and the hyperplane:** From the figure 2, perpendicular distance of the vector space to the origin is $\frac{b}{||w||}$ by using the following procedure:

  1. Scale $w \cdot x + b = 0$ by $\frac{1}{||w||}$.
  2. Use trigonometry to calculate the distance: $x * \cos\theta + y * \sin\theta = \rho$
  3. Then we can rearrange: $\rho = \frac{b}{||w||}$

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

$$\mathbf{w} \cdot \mathbf{x} = -b$$

$$\frac{\mathbf{w} \cdot \mathbf{x}}{||\mathbf{w}||} = \frac{-b}{||\mathbf{w}||}$$

$$\frac{w_1 x_1}{\sqrt{w_1^2 + w_2^2}} + \frac{w_2 x_2}{\sqrt{w_1^2 + w_2^2}} = \frac{-b}{||\mathbf{w}||}$$

We know $x * \cos\theta + y * \sin\theta = \rho$ in polar coordinates where

$$\rho = \frac{-b}{||\mathbf{w}||}$$

$$\cos\theta = \frac{w_1}{\sqrt{w_1^2 + w_2^2}}$$

$$\sin\theta = \frac{w_2}{\sqrt{w_1^2 + w_2^2}}$$

We can check this by

$$cos^2(\theta) + sin^2(\theta) = 1$$

$$\frac{w_1^2}{w_1^2 + w_2^2} + \frac{w_2^2}{w_1^2 + w_2^2} = 1$$

In the polar coordinates, ρ is the distance to the origin, where in this case $\rho = \frac{-b}{||\mathbf{w}||}$.

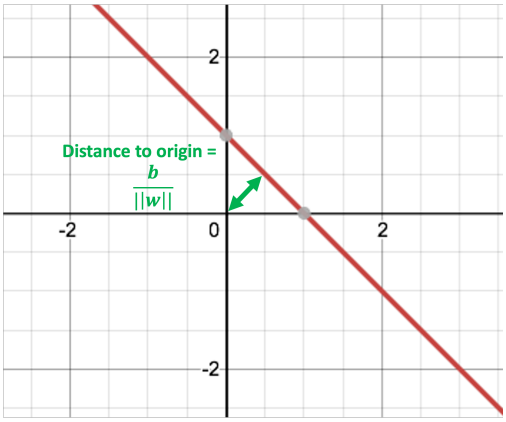

Figure 2: Normal distance between a hyperplane and the origin in 2D

- **Vertical distance between two parallel hyperplanes:** From the figure 3, perpendicular distance between one hyperplane and its adjacent hyperplane is $\frac{1}{||w||}$ assuming that the bias term is 1, by using the following procedures:

  1. We know from the previous example that the perpendicular distance between the hyperplane and the origin is: $\frac{b}{||w||}$
  2. We shift the hyperplane upwards by 1, to $w \cdot x + b = -1$. We rearrange to get $w \cdot x + (b + 1) = 0$. Using the formula for the perpendicular distance between the hyperplane and origin, we get that the distance from this line to the origin is: $\frac{b+1}{||w||}$.
  3. Hence, the pependicular distance between these two hyperplanes is $\frac{b+1}{||w||} - \frac{b}{||w||} = \frac{1}{||w||}$.

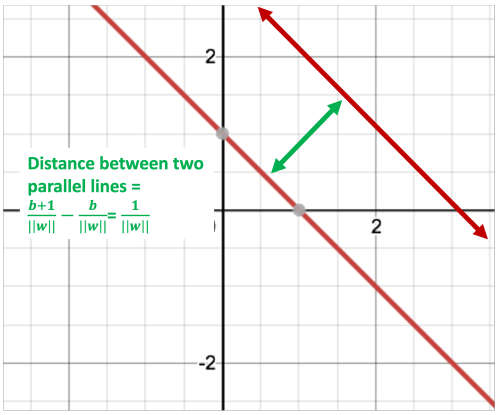

Figure 3: Normal distance between two parallel hyperplanes in 2D

2.2.2 Hyperplanes (lines) in 3D

We can extend this same idea to a higher dimension. For instance,a 3D plane will have the same equation but with different dimensions.The dimensions of a vector in 3D (outside bias form) is 3.

Vertical distance between two parallel hyperplanes: From the figure 4, perpendicular distance between one hyperplane and its adjacent hyperplane is $\frac{2}{||w||}$ assuming that the bias term is at increment of 1(in this case, if we shift the plane in the +1 and -1 directions, the distance between the two of them is $\frac{2}{||w||}$.), by using the following procedures:

1. Hyperplane $(w \cdot x + b = -1 \rightarrow w \cdot x + (b+1) = 0)$ has distance from the origin of: $\frac{b+1}{||w||}$
2. Hyperplane $(w \cdot x + b = 1 \rightarrow w \cdot x + (b-1) = 0)$ has distance from the origin of: $\frac{b-1}{||w||}$
3. Hence, the pependicular distance between these two hyperplanes: $\frac{b+1}{||w||} - \frac{b-1}{||w||} = \frac{2}{||w||}$

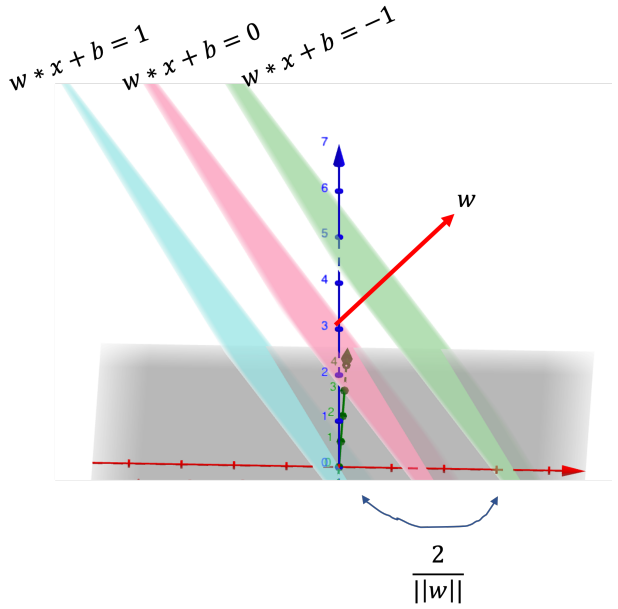

Figure 4: Normal distance between two parallel hyperplanes in 3D

3 Support Vector Machine (Maximum-Margin Classifier)

Consider a binary clasification problem in which we have some dataset $\mathcal{D}$ containing data points of label $y \in \{1, -1\}$. Our task is to learn some decision boundary that best classifies this data. Assume that the data is linearly separable. There is an infinite set of possible separating hyperplanes for linearly separable data, but we want to learn a linear separator with as large of a margin as possible.

Definition 1 (Support Vectors or Interior Points)**.** *A support vector, or interior point, is the data point that lies closest to the hyperplane on each side of the decision boundary.*

Definition 2 (Margin)**.** *A margin (γ) is the distance between an interior point and the separating hyperplane.*

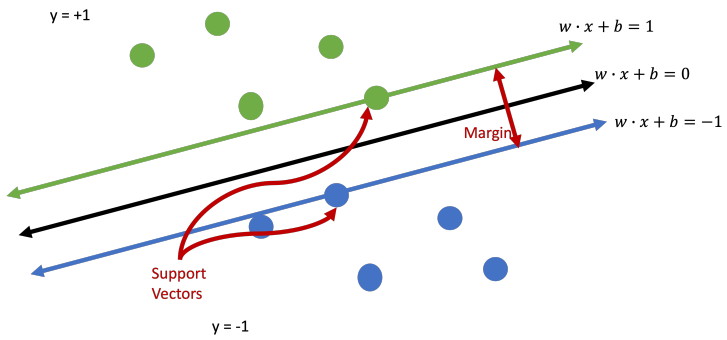

Figure 5: Support vectors are the points closest to the decision boundary, or separating hyperplane. The margin is the distance between support vector and hyperplane.

The objective is to find the separating hyperplane that maximizes the margin, γ so that we can find a hyperplane that can most distinguish and separate the datesets, hence why it is called the Maximum Margin Solution. The maximum-margin classifier is also the most stable hyperplane to perturbations of data, meaning that if you were to move data points around slightly, the hyperplane would most likely remain valid under those small changes. The method is also called a Support Vector Machine (SVM).

3.0.1 Hard SVM

Recall that a hyperplane in 3D can be represented as $\mathbf{w} \cdot \mathbf{x} + b = 0$ whose distance to the origin is $\frac{b}{||w||}$. Shifting this hyperplane up by 1 gives us $\mathbf{w} \cdot \mathbf{x} + b = -1$ and down by 1 gives us $\mathbf{w} \cdot \mathbf{x} + b = 1$. The margin, or gap, between these two parallel hyperplanes is $\frac{b+1}{||w||} - \frac{b-1}{||w||} = \frac{2}{||w||}$. See Figure 5.

Therefore, our objective can be formulated as a maximization problem.

$$\begin{aligned} \max_{\mathbf{w}} \quad & \frac{2}{||\mathbf{w}||} \\ \text{s.t.} \quad & \begin{cases} \mathbf{w} \cdot \mathbf{x_i} + b \geq 1 \text{ if } y_i = 1 \\ \mathbf{w} \cdot \mathbf{x_i} + b \leq 1 \text{ if } y_i = -1 \text{ for } i = 1, ..., N \end{cases} \end{aligned} \tag{1}$$

We can further formulate this as a minimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & ||\mathbf{w}||^2 \\ \text{s.t.} \quad & y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1 \text{ for } i = 1, .., N \end{aligned} \tag{2}$$

By taking the reciprocal of the objective $\frac{||\mathbf{w}||}{2}$, we can change the problem into a minimization problem. We are able to drop the scalar and can add the square to $||\mathbf{w}||^2$, because the minimum point of the objective is the same when squared. The squared objective simply means that we are penalizing large values of $||\mathbf{w}||$ more.

This is a convex quadratic programming (QP) problem, since we are minimizing a quadratic objective with linear constraints. Thus, a unique solution exists to this problem.

This minimization problem is known as the "Primal Formulation" of a linear SVM. The objective function is $\min_{\mathbf{w}} ||\mathbf{w}||^2$. The constraints $y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1$ for $i = 1, .., N$ are known as "hard" constraints, because we assume separability of the data. Hence, this formulation is also known as the "Hard-Margin SVM". If there is noise in the dataset, or the data is non-separable, it may not be possible to satisfy all constraints. In the next section, we relax the linear separability assumption in the "Soft-Margin SVM."

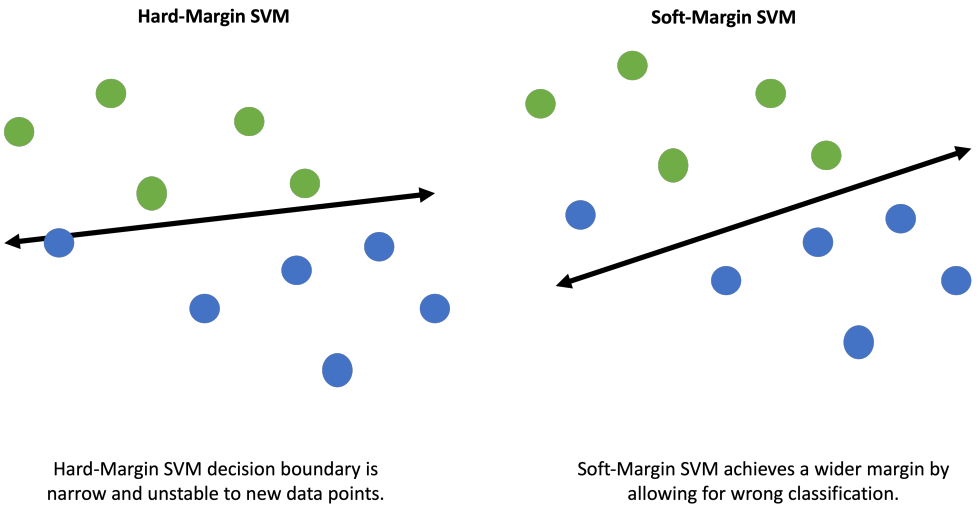

Figure 6: Hard-margin SVM achieves a narrow margin, but Soft-Margin SVM allows for a wider margin to be achieved.

3.1 Soft-Margin SVM

With Soft-Margin SVM, we relax the assumption that the data must be linearly separable. Soft-margin SVM is useful, because most real-world data is noisy, and a hard SVM classification on noisy data can result in very narrow margins that tend to be unstable. By allowing the classifier to make some mistakes, we can get a more stable decision boundary with larger margins. See Figure 7 and 8.

3.1.1 Slack Variable

Soft-Margin SVM introduces a slack constraint, which represents how mistake-tolerant the soft-margin SVM is. The slack variable, ξ_i, geometrically, is the distance from a misclassified point to the margin corresponding to its true label. $\xi_i \geq 0$.

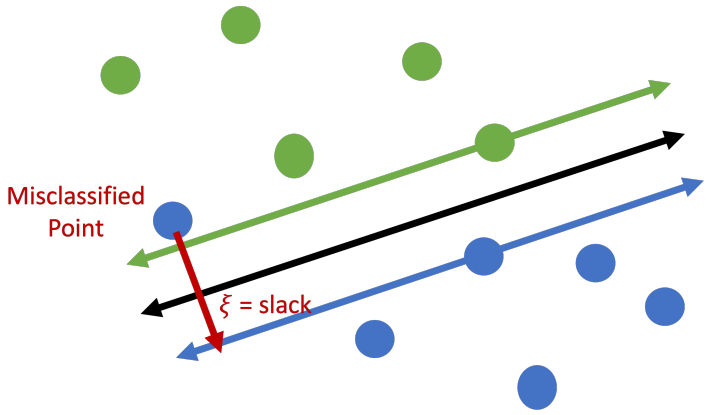

Figure 7: The slack constraint allows the Soft-Margin SVM to make mistakes in order to achieve a larger margin. The constraint allows the algorithm to trade off between optimizing for a larger margin or more accurate classification.

In general, we want a reasonable small slack variable. A small ξ_i allows for small mistakes to be made by the algorithm. Small mistakes are mistakes close to the decision boundary, and not far from the hyperplane corresponding to the true label. A large ξ_i means that the SVM is allowing for large mistakes to be made.

3.1.2 Soft-Margin SVM Objective

In order to allow for some misclassification of the SVM, we can add $-\xi_i$ to the original hard-margin constraint, $y_i(w \cdot x_i + b) \geq 1$. The constraint becomes $y_i(w \cdot x_i + b) \geq 1 - \xi_i$.

The Soft-Margin problem is thus formulated as:

$$\begin{aligned} \min_{\mathbf{w}} \quad & ||\mathbf{w}||^2 + C \sum_i \xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1 - \xi_i \text{ for } i = 1, .., N \end{aligned} \tag{3}$$

Important notes on the soft-margin objective:

- The slack variable allows for mistakes, as long as the inverse margin is minimized.
- Every constraint can be satisfied if slack is large.
- C is a regularization parameter. If C is small, we do not prioritize keeping the slack small, and thus this results in large margins. If C is big, we prioritize keeping the slack small, and thus, we get a small margin.
- This is still a quadratic programming (QP) problem since it still has a quadratic objective and linear constraints. Thus, we know that there exists a unique solution.

3.2 Stochastic (Sub) Gradient Descent for Soft-Margin SVM Classifier

In order to solve the Soft-Margin SVM, we will merge the linear constraints into the objective function. We merge the bias term, b, into the weight vector, $\mathbf{w}$. We move the slack variable to one side of the optimization constraint inequality, and get $1 - y_i(\mathbf{w} \cdot \mathbf{x_i}) \leq \xi_i$. This term is then added to the objective, giving us

$$\min_{\mathbf{w}} \quad \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{N}\sum_{i=1}^{N} 1 - y_i \mathbf{w}^T \mathbf{x}_i \tag{4}$$

The problem with this objective function is that the component $1 - y_i \mathbf{w}^T \mathbf{x}_i$ becomes is a very large negative value for "easy" data points that are correctly classified by a large margin, meaning the data points lie far from the separating hyperplane. For mistaken points, the value $1 - y_i \mathbf{w}^T \mathbf{x}_i$ is small, and is thus overpowered by large positive values of correct points.

When the SVM classifies a data point "very" correctly, the slack variable is close to negative infinity. When it's highly mistaken, the slack variable is close to positive infinity. The portion of the spectrum we care about is the "slightly correct" to "very mistaken" part of the spectrum because we want to penalize mistakes and "weakly correct" data points to better optimize the parameters. The SVM should ignore "very correct" data because we don't want those points to influence the weight, $\mathbf{w}$ too much since they are already very correct. In other words, we don't want the correct points resulting in very large negative values to wash out the positive values from the mistaken points.

We use the Hinge loss to resolve this problem so that we only penalize for mistakes and weakly correct points.

$$\min_{\mathbf{w}} \quad \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{N}\sum_{i=1}^{N} \max\{0, 1 - y_i \mathbf{w}^T \mathbf{x}_i\} \tag{5}$$

The regularization term $\frac{\lambda}{2}||w||^2$ is quadratic. The loss function is linear $\frac{1}{N}\sum_i^N max\{0, 1 - y_i w^T x_i\}$. Therefore, this function is convex. **This is Online Mirror Descent with quadratic regularization and a piece-wise linear loss function.**

Several Important Keypoints:

- The function is convex because it resembles the online mirror descent form with quadratic regularization and a piece-wise linear loss function.
- The function is not differentiable because the hinge has a kink point.
- We can not use gradient descent to solve the loss function because the function is not differentiable.
- We could use sub-gradient descent to optimize the function.

3.3 Sub-gradients

For a differentiable convex function, there's only one line that lower bound the function at any point. For a non-differentiable convex function, there's many (infinite) lines that lower bound the function. These are called **subgradients**.

The hinge loss function is not differentiable. For the hinge loss function, there are many possible sub-gradients. We will use two subgradients:

$$z_i = \begin{cases} 0 & \text{if } y_i \mathbf{w}^T \mathbf{x}_i \\ -y_i \mathbf{x_i} & \text{otherwise} \end{cases} \tag{6}$$

This piecewise subgradient states that if the SVM and observation label are in agreement, meaning there is sufficient distance from the hyperplane $\rightarrow y_i \mathbf{w}^T \mathbf{x}_i \geq 1$, then the gradient is 0. If the SVM classification is wrong, or is not sufficiently correct represented by $y_i \mathbf{w}^T \mathbf{x}_i \leq 1$, then the gradient is $-y_i \mathbf{x_i}$.

Algorithm 3 Soft SVM

1: $\mathbf{w}^{(0)} \leftarrow 0 \in R^N$ {Written in OMD format}
2: **for** $t = 0, \cdots, T-1$ **do**
3: RECEIVE $(\mathbf{y}_d, x_d \sim D)$ {Receive sample from environment (training data)}
4: $\boldsymbol{\theta}^t = \boldsymbol{\theta}^{(t-1)} + y_d x_d \cdot \mathbb{1}[y_d(\mathbf{w}^{(t)} \mathbf{x}_d) < 1]$ {Dual parameter update}
5: $\mathbf{w}^{(t+1)} \leftarrow \frac{1}{\lambda(t+1)} \boldsymbol{\theta}^t$ {Mirror projection}
6: **end for**

We sample an observation-label pair from the dataset. Another way of thinking of this is receiving a data sample from an environment. We then perform the dual parameter update. The new parameter is the old parameter plus the gradient of the loss function. Next, we perform the mirror projection.

3.4 Comparison of SoftSVM with Perceptron

Recall the Perceptron algorithm written in the online mirror descent format.

Algorithm 4 Perceptron Algorithm

1: $\mathbf{w}^{(0)} \leftarrow 0 \in R^N$ {Written in OMD format}
2: **for** $t = 0, \cdots, T-1$ **do**
3: RECEIVE $(\mathbf{x}^{(t)}, y^{(t)})$
4: $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} + y^{(t)} \mathbf{x}^{(t)} \cdot \mathbb{1}[y^{(t)} \langle \mathbf{w}^{(t)}, \mathbf{x}^{(t)} < 0 \rangle]$ {Dual parameter update}
5: $\mathbf{w}^{(t+1)} = \theta^{(t)}$ {Mirror projection}
6: **end for**

SoftSVM and Perceptron both have piecewise loss functions due to both using hinge loss and quadratic regularization. In terms of the algorithm itself, SVM and Perceptron algorithm have a similar dual parameter update. SVM uses a soft margin, but Perceptron does not use a margin. As a result, the Perceptron algorithm does not try to push the data points away from a hyperplane, but simply says that data must lie on one side of the hyperplane. The two algorithms also have a similar mirror function.

4 Summary

In this lecture, we covered hyperplanes and the geometry of a linear classifier. The SVM algorithm tries to classify all of its datapoints on the correct side of a separating hyperplane while maximizing the margin between the data and the hyperplane. Soft-margin SVM relaxes the assumption of linear separability and allows for a tradeoff between classification accuracy and margin width. Soft-margin SVM can be solved with Online Sub-gradient Descent.

5 Appendix

5.1 Kernels and Non Linear SVM

In the class, we mostly explored the case of linear separable datasets. While soft SVM could handle some non-linear datasets with some outliers, it usually performs very poorly when the datasets are strictly non-linear. For example, in figure 8, we can see that the SVM algorithm performs very poorly on this dataset.

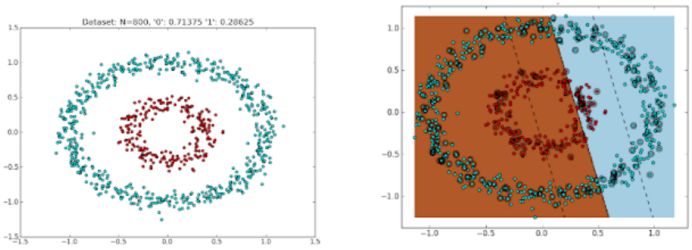

Figure 8: Linear SVM on non-linear datasets.

In general, there are two methods to resolve this problem. (1) project the dataset to higher dimension or (2) use an non-linear kernel

5.1.1 Higher dimension projection

For instance, if we have the original dataset in 2D, we could perform the mapping $X^2 \rightarrow \{x^3, x^2\}$. Then the dataset might be separable in this new 3D space. As we see in the figure 9, the data is non-linear in 2D but can be linearly separated in 3D.
Note: This will greatly increase the computation cost. As our datasets are getting more complex, projecting such datasets into higher dimension might not be computationally feasible

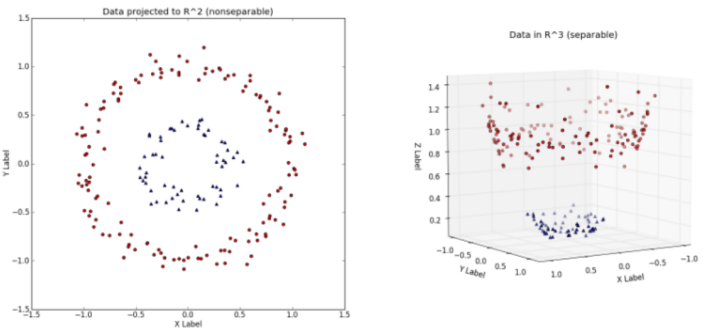

Figure 9: Data projection into a higher dimension.

5.2 Kernel tricks

Kernels are essentially the functions that transform the features and weights into a way to split the data (How similar they are) or in other words, they are the tools that SVM uses to reason about the dataset

Common kernels are linear(which is the one we used), non-linera kernel, polynominal kernel, sigmod kernel, etc. For instance, if we apply non linear kernel as seen in figure 10, we could get non-linear hyperplanes that could better separate the dataset

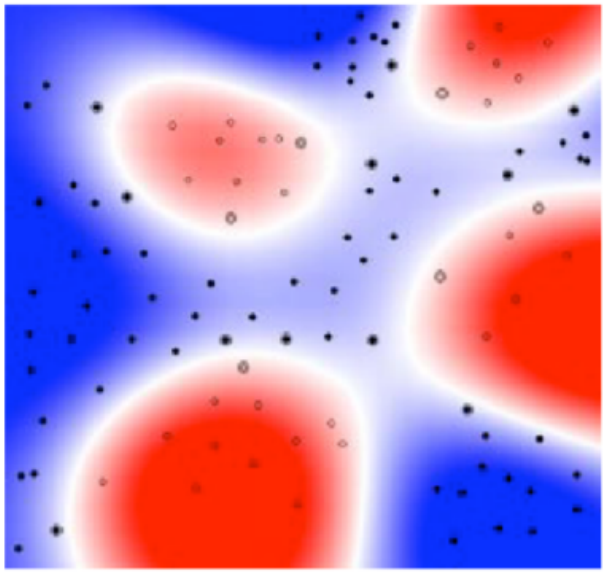

Figure 10: Nonlinear rbf Kernel

References

[1] Andrew Ng. CS229 Lecture Notes, Stanford. https://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf

[2] Tendolkar, G. (n.d.). 2.1.6 kernels and non linear SVM - Machine Learning Notebook. Google Sites. Retrieved February 22, 2022, from https://sites.google.com/site/machinelearningnotebook2/classification/bi classification/kernels-and-non-linear-svm

[3] R. Berwick, SVM Lecture Notes, MIT .http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf

[4] Huh, Myung-Hoe. (2015). Kernel-Trick Regression and Classification. Communications for Statistical Applications and Methods. 22. 201-207. 10.5351/CSAM.2015.22.2.201.

[5] Hofmann, Martin. "Support Vector Machines — Kernels and the Kernel Trick An elaboration for the Hauptseminar " Reading Club : Support Vector Machines "." (2006).