Margin-based Classification

Machine Learning Course (BITS F464)

Topics

This lecture consists of the following topics:

- Linear separability
- Linear classifiers
- Support Vector Machines (SVM)
- Kernels

Linear separability I

Consider a binary classification problem:

Given training data $(\mathbf{x}_i, y_i)$ for $i = 1 \ldots N$, ith d-dimensional input $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$.

We learn a classifier $f(\mathbf{x})$ such that

$$f(\mathbf{x}_i) \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

Alternatively, we can write this as $y_i f(\mathbf{x}_i) > 0$ for a correct classification rule.

Linear separability II

The following figure shows the distribution of data points for a two dimensional classification problem. You could consider the blue (circular) dots as class-1, and the red (triangular) dots are class-2.

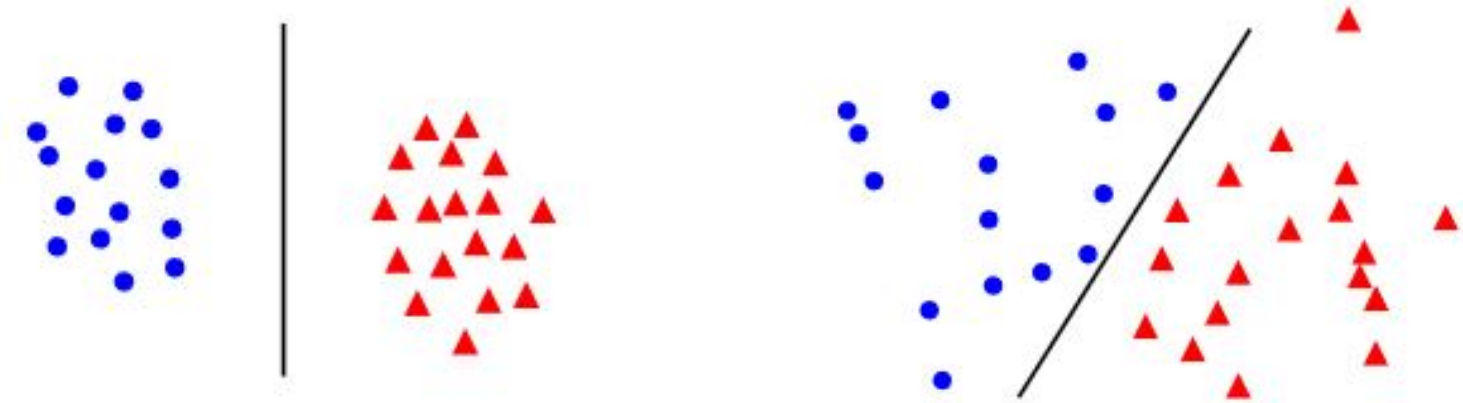
(a) Problem 1

(b) Problem 2

Figure 1: Two different binary classification problems

Linear separability III

A linear decision boundary is easy to see for the problem shown in Fig. (1a). Two example cases are:



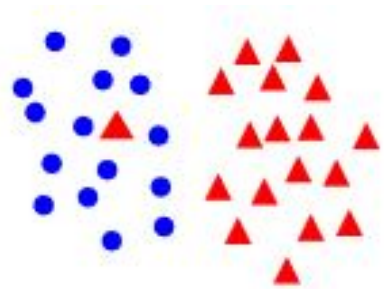
(a) Problem 1 linearly separable

(b) Problem 2 linearly separable

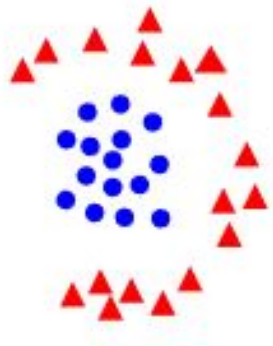
Figure 2: Problems with achievable linear decision boundary

Linear separability IV

However, achieving a well-separating decision boundary for the problem in Fig. (1b) is not possible.



(a) Problem with a single point in other group



(b) Problem with non-linearly separable data points

Figure 3: Problems which are NOT linearly separable

Pay attention to the left hand side figure. Is it too much different from the ones we saw in previous slide?

Linear classifier I

Now we learn a classifier, mathematically written as $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$.

In two dimension, the decision boundary (also called discriminant) is a line; $\mathbf{w}$ is called the weight vector, and b is the bias, which is responsible for affine transformation in the domain.

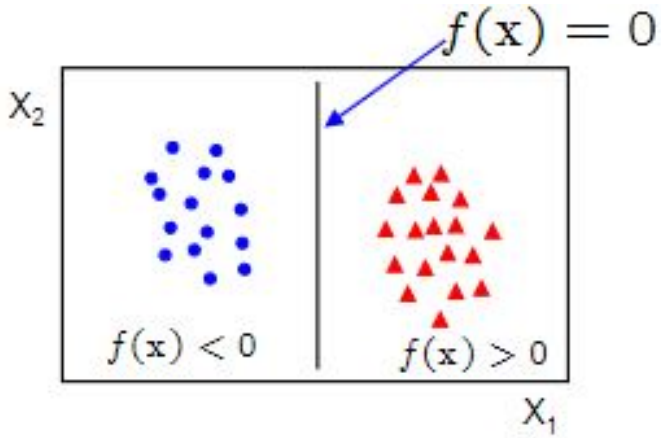


Figure 4: Linear classifier $f(\mathbf{x}) = 0$ for the problem 1.

Linear classifier II

In three dimension, the boundary is a plane, and higher dimensions it is called a hyperplane.

Note:

- For a linear classifier, the primary objective is to efficiently learn the parameters $\mathbf{w}$ and b given the training data of the form described earlier. The training data then can be discarded. For future classification (test), these weight and bias parameters are sufficient for deciding a class.
- On the contrary, this is not true for distance based machine learning models such as k-nearest neighbor or its weighted version.

Revisiting Perceptron I

Given linearly separable data, $\mathbf{x}_i$ labeled into two categories $y_i = \{-1, +1\}$, find a weight vector $\mathbf{w}$ such that the discriminant function

$$f(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x}_i + b$$

that separates each data point $\mathbf{x}_i$ for $i = 1 \ldots N$.

Now we look at the following learning procedure (algorithm) that find such a hyperplane (we consider that $\mathbf{x}_i$ is multidimensional).

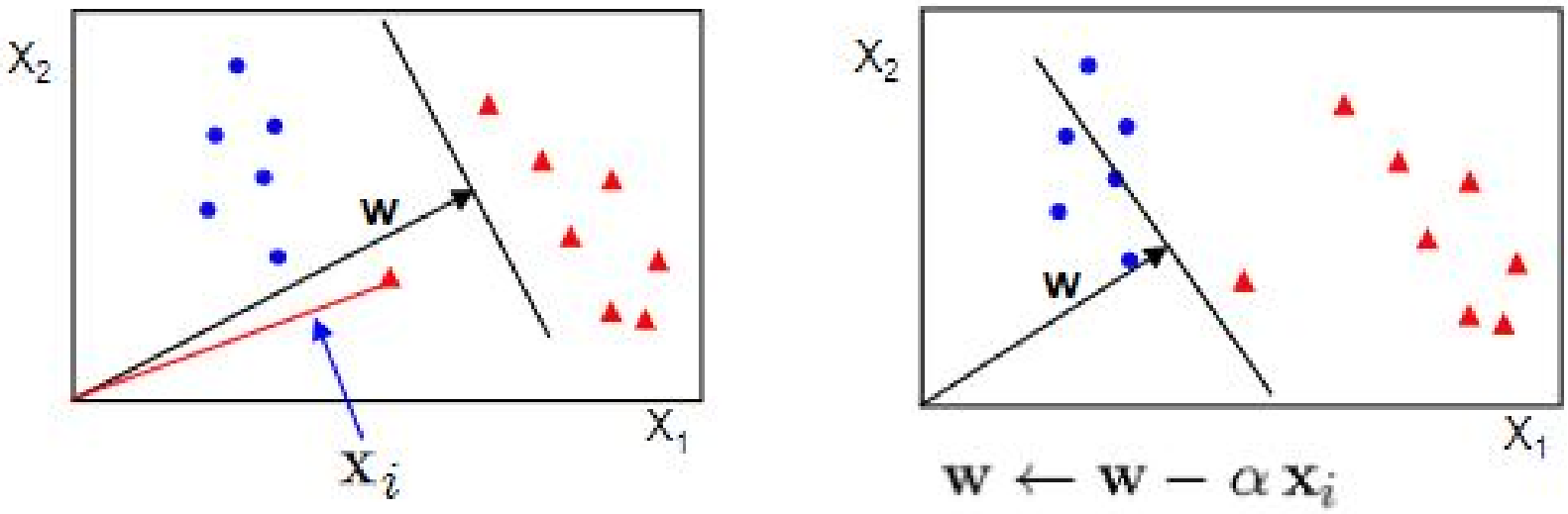
Revisiting Perceptron II

The perceptron learning algorithm: We write the classifier as $f(\mathbf{x}_i) = \mathbf{w}_o^\top \mathbf{x}_i + w_o = \mathbf{w}_o^\top \mathbf{x}_i$, where bias is denoted as w_0 and has been included in the weight vector itself. Therefore, the input can be written as $\mathbf{x}_i = (\mathbf{x}_i, +1)$.

1. Initialize $\mathbf{w} = 0$
2. Draw a data point $\{\mathbf{x}_i, y_i\}$ from the given training set
   2.1 If $\mathbf{x}_i$ is misclassified, then $\mathbf{w} \leftarrow \mathbf{w} + \alpha \text{ sign}(f(\mathbf{x}))\ x_i$
3. Repeat step 2–3, until all the data is correctly classified.

The final weight vector is written as $\mathbf{w}_o$ (o : optimal).

Revisiting Perceptron III



(a) Before update

(b) After update

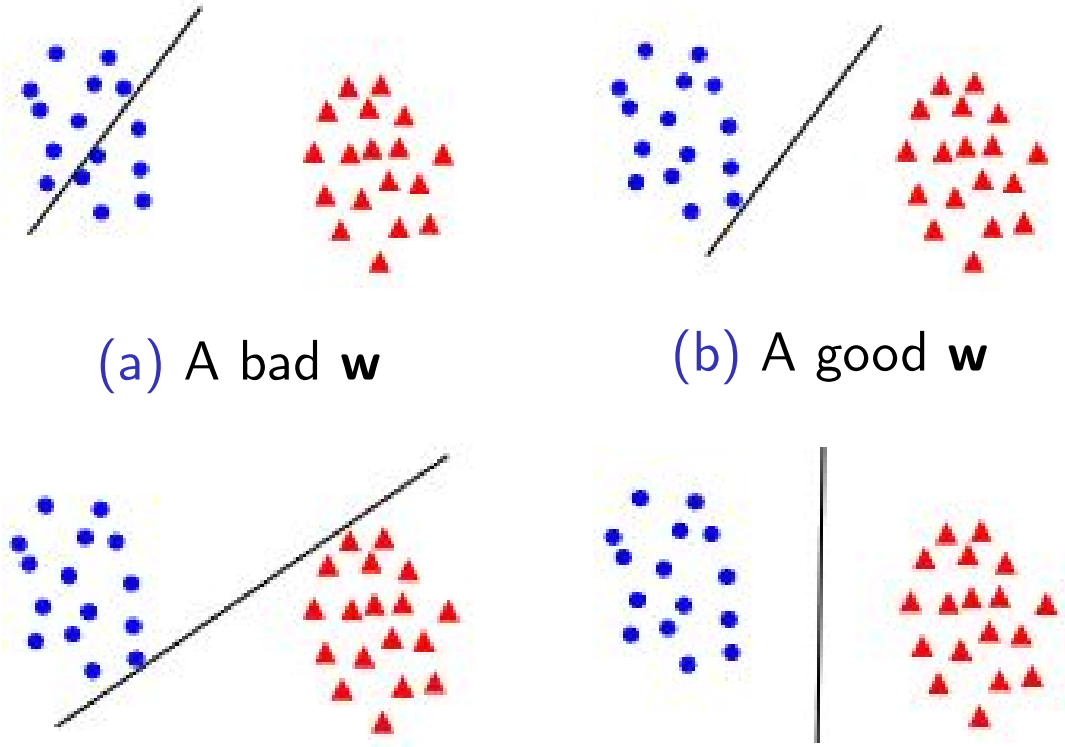
Figure 5: Perceptron boundary: before and after update using the learning rule

Revisiting Perceptron IV

Convergence It can be mathematically proved that if the training data is linearly separable, then perceptron can converge to an optimal state after finite iterations. Convergence of the perceptron can be understood as a state after which learning starts saturating i.e. $\mathbf{w}(t) = \mathbf{w}(t+1) = \mathbf{w}(t+2) = \ldots$.

Revisiting Perceptron V

But, which hyperplane should be chosen?



(a) A bad **w**

(b) A good **w**

(c) Another good **w**

(d) A better **w**

Figure 6: Perceptron boundary: before and after update using the learning rule

Revisiting Perceptron VI

Did you chose this?

- Fig (6d) (the one in the last quadrant)

How did you chose it?

- You looked at the distance of the hyperplane from data points present in both of its sides.
- Fig. (6d) justifies such a distance, whereas others do not.
- In fact, a slightly deviating data point in the test set could be misclassified by the hyperplanes shown in Fig. (6b) or Fig. (6c).

Revisiting Perceptron VII

Note Sometimes, we can have a logistic function which is suitable as a classifier and adds a fair amount of flexibility to the decision boundary. An example is

$$f(\mathbf{x}_i) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}_i}}$$

Based on the domain of y_i, we can also have

$$f(\mathbf{x}_i) = \frac{1 - e^{-\mathbf{w}^\top \mathbf{x}_i}}{1 + e^{-\mathbf{w}^\top \mathbf{x}_i}}$$

Note Perceptron inherently learns with a cost(loss) function based on the error between $f(\mathbf{x}_i)$ and y_i.

Revisiting Perceptron VIII

Margin based loss:

- The distance between data points from one class to another class is called the **margin**.
- Margin is most stable under perturbations of the inputs. So, instead of learning a hyperplane, SVM learns a margin along with the hyperplane.

Support Vector Machine (SVM) I

Consider a linearly separable case (later it will be relaxed) of classification problem.

Let Δ denotes the minimal distance between a hyperplane $f(\mathbf{x}_i)$ and a data point $\mathbf{x}_i$.

So, $f(\mathbf{x}_i)$ with margin 2Δ satisfies the following constraints:

$$\mathbf{w}^\top \mathbf{x}_i + b \geq +\Delta; \text{if } y_i = +1$$

and

$$\mathbf{w}^\top \mathbf{x}_i + b \leq -\Delta; \text{if } y_i = -1$$

Support Vector Machine (SVM) II

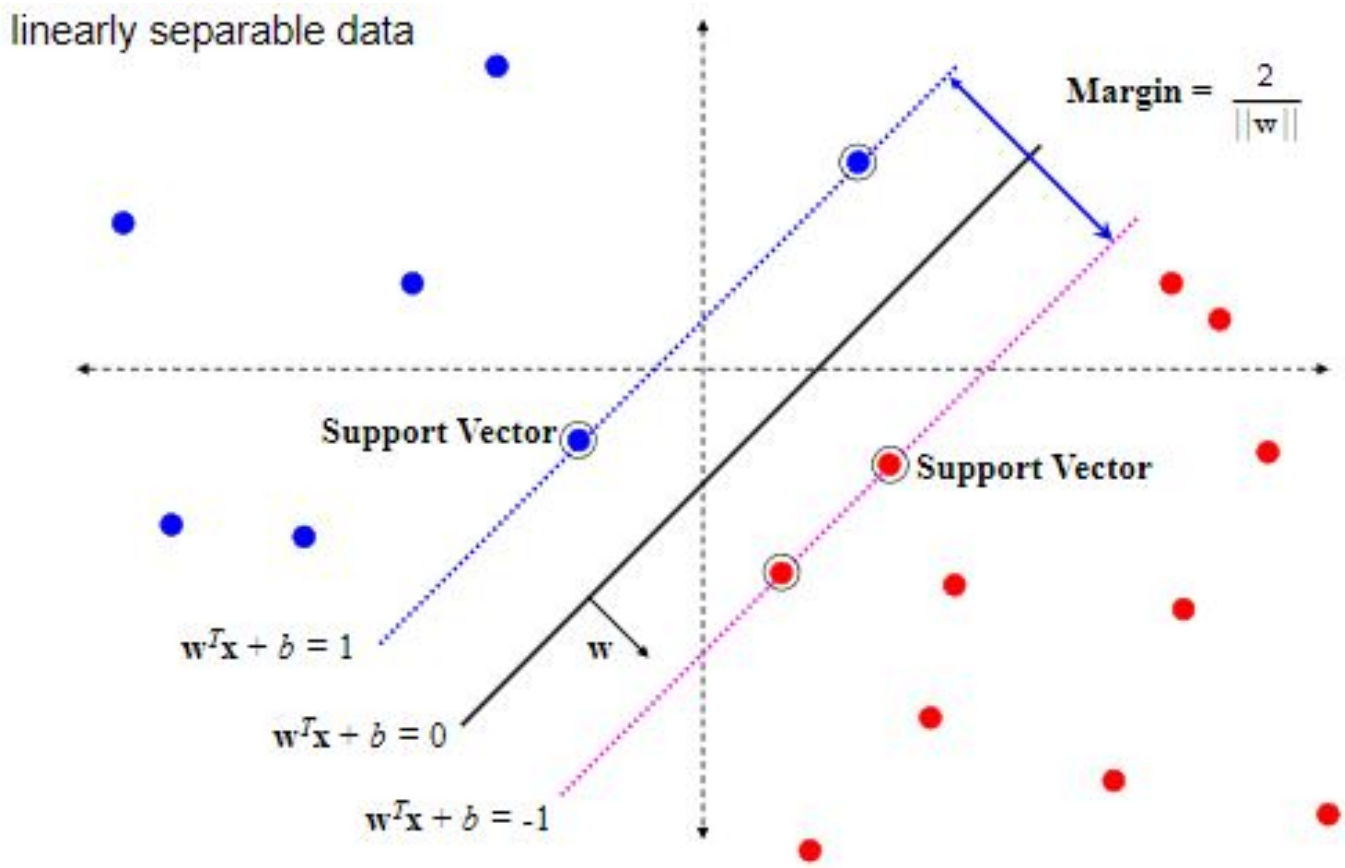


Figure 7: Margin for linear separable case

Support Vector Machine (SVM) III

Understanding Fig. (7): The distance between the hyperplane

$$f(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x}_i + b = 0$$

and a sample $\mathbf{x}_i$ is given as

$$\frac{f(\mathbf{x}_i)}{||\mathbf{w}||} = \frac{\mathbf{w}^\top \mathbf{x}_i + b}{||\mathbf{w}||}$$

For a margin of 2Δ, all the data points are at least Δ away from the hyperplane and must satisfy the inequality

$$\frac{y_i\{\mathbf{w}^\top \mathbf{x}_i + b\}}{||\mathbf{w}||} \geq \Delta; i = 1 \ldots N$$

Or,

$$y_i\{\mathbf{w}^\top \mathbf{x}_i + b\} \geq \Delta||\mathbf{w}||; i = 1 \ldots N$$

Support Vector Machine (SVM) IV

Rescaling, the parameter $\mathbf{w}$ and b by fixing the scale $\Delta||\mathbf{w}|| = 1$ i.e. $(\Delta = \frac{1}{||\mathbf{w}||})$ leads to the canonical form representation of the separating hyperplane.

This is equivalent to choosing a normalization such that $\mathbf{w}^\top \mathbf{x}_+ + b = +1$ and $\mathbf{w}^\top \mathbf{x}_- + b = -1$ for positive and negative support vectors respectively.

Then the margin is given by the distance between the support vectors in both the classes.

$$\frac{\mathbf{w}}{||\mathbf{w}||}(\mathbf{x}_+ - \mathbf{x}_-) = \frac{1-(-1)}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$$

SVM – Optimization I

Learning the SVM can be formulated as an optimization problem

$$\max_{\mathbf{w}} \frac{2}{||\mathbf{w}||}$$

subject to constraints

$$y_i\{\mathbf{w}^\top \mathbf{x}_i + b\} \geq 1; \text{ for } i = 1 \ldots N$$

Or, equivalently

$$\min_{\mathbf{w}} ||\mathbf{w}||^2$$

subject to constraints

$$y_i\{\mathbf{w}^\top \mathbf{x}_i + b\} \geq 1; \text{ for } i = 1 \ldots N$$

SVM – Optimization II

This is essentially a quadratic optimization problem subject to linear constraints and therefore there is existence of unique local minimum. *That is the objective function is convex*.

Solution: We can use Lagrangian optimization method where the objective function can be converted to a form $f - \lambda \times \textit{constraints}$.

So, as per Lagrangian method of optimization, we minimize the objective function by incorporating the constraints $y_i\{\mathbf{w}^\top \mathbf{x}_i + b\} - 1 \geq 0$; for $i = 1 \ldots N$:

$$\mathcal{L} : ||\mathbf{w}||^2 - \sum_{i}^{N} \lambda_i y_i \{\mathbf{w}^\top \mathbf{x}_i + b\} - 1$$

SVM – Optimization III

The roots of the above objective function equation be obtained by solving the following equations:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0$$

and

$$\frac{\partial \mathcal{L}}{\partial \lambda_i} = 0; \text{ for } i = 1 \ldots N$$

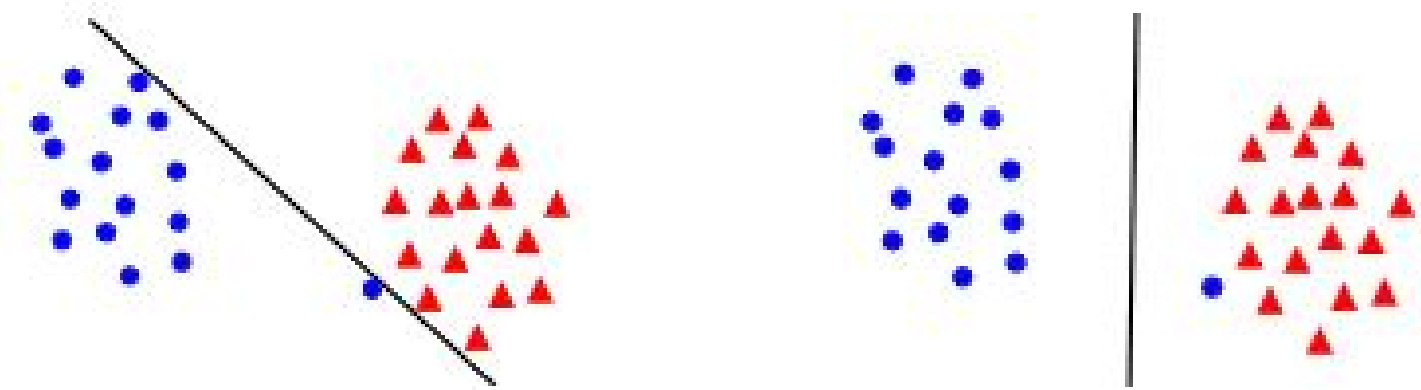
The parameters λ_is are called the Lagrangian multipliers.

SVM – Optimization IV

- Let us now look at the problem of linear separability.
- Our focus would now be on the violation of constraints in objective function.
- In fact, we would ask the question, is it a good idea to obtain a large margin even though a few constraints are violated?
- There is a trade off between the margin and the number of mistakes on the training data.

SVM – Optimization V

Look at the following figures.



(a) Narrow margin (no violation)

(b) Large margin (one violation)

Figure 8: SVM — violation and non-violation of constraints and how it is related to margin

Constraint violation: Introducing slack variables I

Fig. (9) clearly shows the effects on learning after introduction of slack variables ξs ($\xi > 0$).

- If $\xi_i \in (0, 1]$, then the point $\mathbf{x}_i$ is between margin and correct side of the hyperplane.
- This is called margin violation. If $\xi_i > 1$, then the point is misclassified.

Constraint violation: Introducing slack variables II

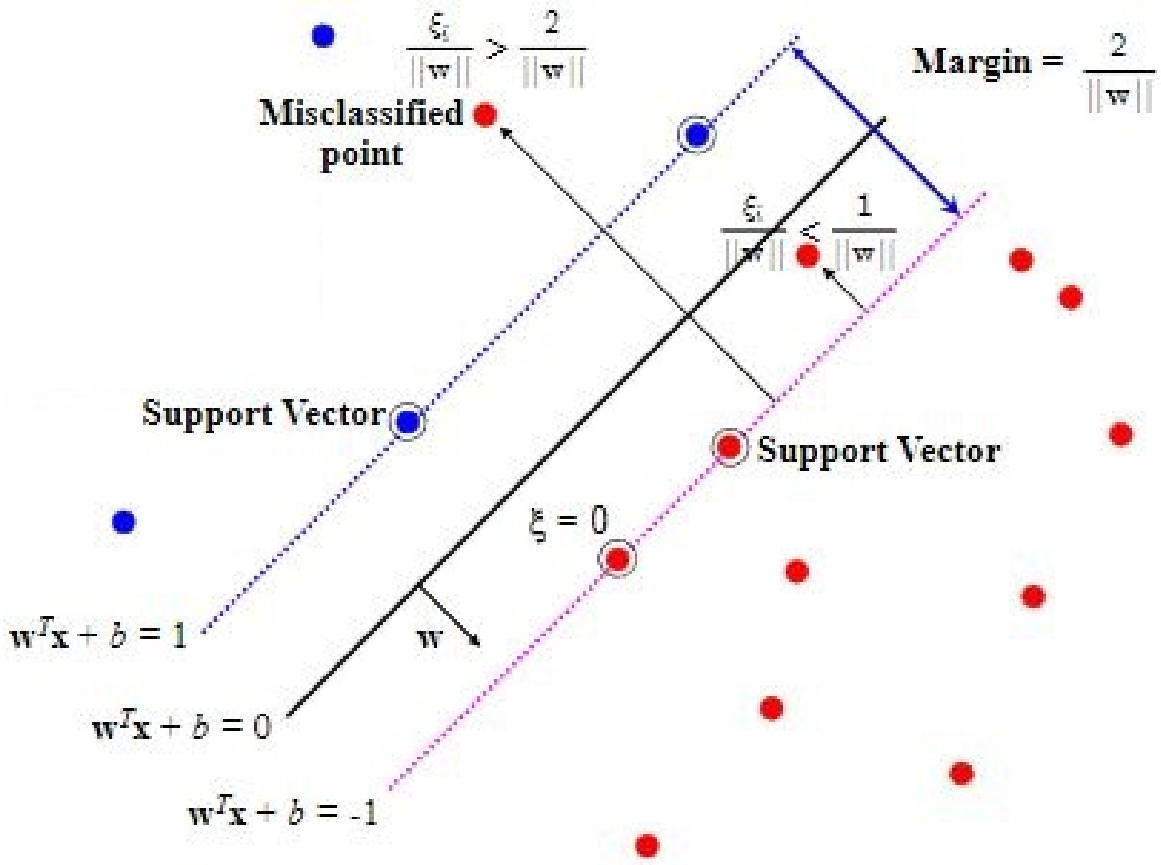


Figure 9: Introduction of slack variables

Constraint violation: Introducing slack variables III

The corresponding objective function for the SVM is

$$\min_{\mathbf{w}\in\mathbb{R}^d,\xi\in\mathbb{R}^+} ||\mathbf{w}||^2 + C\sum_{i}^{N}\xi_i$$

subject to

$$y_i\{\mathbf{w}^\top\mathbf{x}_i + b\} \geq 1 - \xi_i;\ \text{for } i = 1\ldots N$$

Constraint violation: Introducing slack variables IV

Let $\mathcal{M} = \sum_i^N \xi_i$. Here are the properties of C and $\mathcal{M}$.

- If $\mathcal{M}$ is very large, then there are many violations. However, every constraint can be satisfied. A large margin can be achieved.
- C can be considered as a regularization parameter. Small C allows constraints to be easily ignored, thereby introducing large margin.
- Large C does not allow the constraints to be ignored. So, we get narrow margin.
- $C = \infty$ allows none of the constraints to be ignored. We achieve a large margin.

Constraint violation: Introducing slack variables V

Since, C is a constant, the objective function is still convex and can be solved using Lagrangian optimization described in one of the earlier sections.

Relation between ξ and $f(\mathbf{x}_i)$ I

Rewriting the constraints in terms of $f(\mathbf{x}_i)$, we get

$$y_i f(\mathbf{x}_i) \geq 1 - \xi_i$$

Therefore,

$$\xi_i = 1 - y_i f(\mathbf{x}_i)$$

Since, $\xi_i > 0$, we can write

$$\xi_i = max(0, 1 - y_i f(\mathbf{x}_i))$$

Hence, the learning problem is equivalent to the unconstrained optimization problem over $\mathbf{w}$.

$$\min_{\mathbf{w} \in \mathbb{R}^d} ||\mathbf{w}||^2 + C \sum_{i}^{N} \max(0, 1 - y_i f(\mathbf{x}_i))$$

Relation between ξ and $f(\mathbf{x}_i)$ II

The above equation has a very interesting property. The second term $\max(0, 1 - y_i f(\mathbf{x}_i))$ is in fact the loss function, and the first term $||\mathbf{w}||^2$ is the regularization term.

Relation between ξ and $f(\mathbf{x}_i)$ III

The loss function discussed above i.e. $\max(0, 1 - y_i f(\mathbf{x}_i))$ is called the **hinge loss**, which is an approximation to the 0–1 loss function.

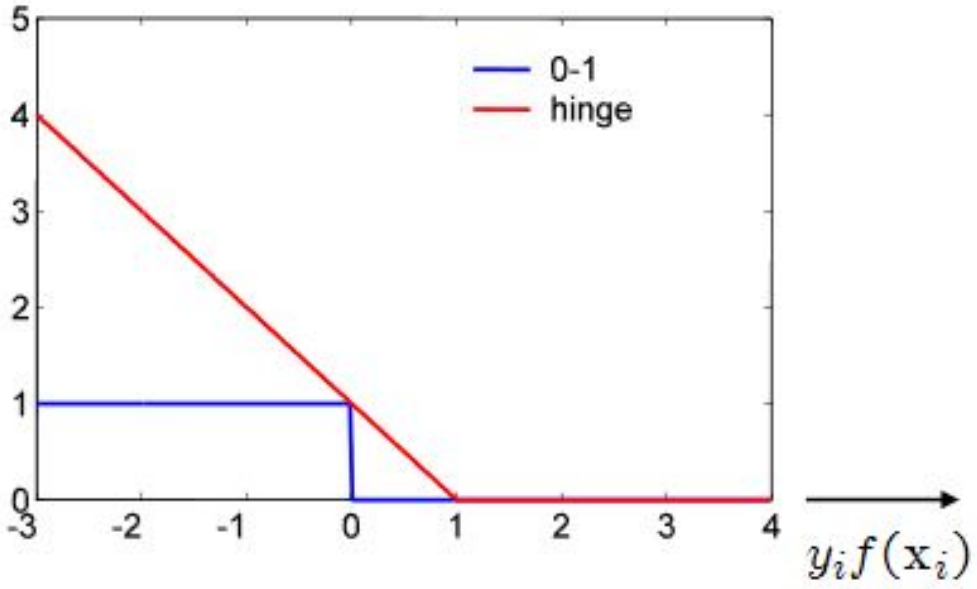


Figure 10: Hinge loss

Relation between ξ and $f(\mathbf{x}_i)$ IV

- If we closely look again at the objective function that includes the hinge loss, we see that the regularization term is convex.
- The second term also is convex. Therefore, we can say that the whole term $\mathbb{R}^d||\mathbf{w}||^2 + C\sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i))$ is convex and has a unique solution.
- We now look at the minimization of the cost function using stochastic gradient descent (SGD) procedure.

Minimization of $\mathbb{R}^d||\mathbf{w}||^2 + C\sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i))$ I

The iterative weight update equation based on the SGD procedure is given as

$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) - \eta(t)\nabla_{\mathbf{w}} \left(\mathbb{R}^d||\mathbf{w}||^2 + C\sum_{i}^{N} \max(0, 1 - y_i f(\mathbf{x}_i)) \right)$$

where $\eta(t)$ is the learning rate and presently shown as a function of time.

Minimization of $\mathbb{R}^d||\mathbf{w}||^2 + C\sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i))$ II

The hinge loss is not differentiable, we can use a sub-gradient:

$$\nabla_{\mathbf{w}} \max(0, 1 - y_i f(\mathbf{x}_i)) = \begin{cases} 0 & y_i f(\mathbf{x}_i) > 1 \\ -y_i \mathbf{x}_i & \text{otherwise} \end{cases}$$

Minimization of $\mathbb{R}^d||\mathbf{w}||^2 + C\sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i))$ III

Now the iterative weight update equation is

$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) - \eta(t)\left(\nabla_{\mathbf{w}}\mathbb{R}^d||\mathbf{w}||^2 + \nabla_{\mathbf{w}} C\sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i))\right)$$

or

$$\mathbf{w}(t+1) \leftarrow \begin{cases} \mathbf{w}(t) - \eta(t)(\lambda\mathbf{w}(t) - y_i\mathbf{x}_i) & \text{if } y_i f(\mathbf{x}_i) < 1 \\ \mathbf{w}(t) - \eta(t)\lambda\mathbf{w}(t) & \text{otherwise} \end{cases}$$

Minimization of $\mathbb{R}^d ||\mathbf{w}||^2 + C \sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i))$ IV

The above equation is based on a slightly modified objective function

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{\lambda}{2} ||\mathbf{w}||^2 + C \sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i))$$

The parameter λ can be considered as the regularization parameter.

Kernels I

- It has been seen that straight forward machine learning model such as a perceptron is unable to learn problems which are not linearly separable.
- It is indeed the case in our studies in SVM as well. Our earlier discussions focused on much on the idea of linear separability of the data points.
- Such a condition (or demand) on the real-world problem is not feasible always.
- There are problems, though simple, can not be learned efficiently by SVM with linear boundary.

Kernels II

For example, an exclusive-or problem.

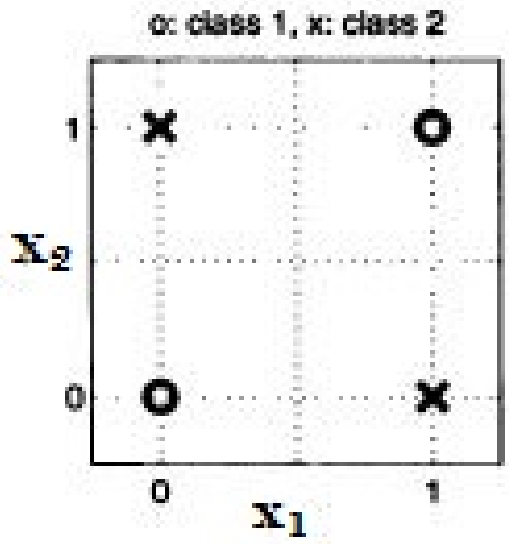


Figure 11: 2-bit XOR problem

Kernels III

We intend to obtain a new set of feature based on the mapping:

$$\mathbf{x} \xrightarrow{\phi(\cdot)} \phi(\mathbf{x})$$

The feature transformation function $\phi(\cdot)$ transforms the original feature space to a higher dimensional feature space, that makes more flexibility to the design of a margin based decision boundary.

The function $\phi(\cdot)$ is called *kernel* and the process of learning on the transformed feature space is called *kernel trick*.

Kernels IV

Some examples of kernel functions:

- **Linear Kernel** This kernel is useful when data points are separable linearly:

$$\phi(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$$

- **Polynomial Kernel** Kernel is a d-degree polynomial:

$$\phi(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + r)^d$$

- **RBF Kernel** This is similar to what we have already seen in RBF-Nets:

$$\phi(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2\right)$$

New Objective Function I

The new objective function for the SVM can be written as

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{\lambda}{2}||\mathbf{w}||^2 + C\sum_{i}^{N} \max(0, 1 - y_i f(\phi(\mathbf{x}_i)))$$

And the iterative weight update algorithm is

$$\mathbf{w}(t+1) \leftarrow \begin{cases} \mathbf{w}(t) - \eta(t)(\lambda\mathbf{w}(t) - y_i\phi(\mathbf{x}_i)) & \text{if } y_i f(\phi(\mathbf{x}_i)) < 1 \\ \mathbf{w}(t) - \eta(t)\lambda\mathbf{w}(t) & \text{otherwise} \end{cases}$$

Further readings

Christopher M. Bishop, "Pattern Recognition and Machine Learning", Springer (2006), ISBN 0-38-731073-8. (Chapter 6 and chapter 7)

Burges, Christopher JC. "A tutorial on support vector machines for pattern recognition", Data mining and knowledge discovery 2, no. 2 (1998): 121-167.