

Machine learning

Linear & nonlinear classifiers

Hamid Beigy

Sharif University of Technology

April 16, 2023





1. Introduction
2. Linear classifiers
3. Perceptron algorithm
4. Support vector machines
5. Lagrangian optimization
6. Support vector machines (cont.)
7. Non-linear support vector machine
8. Linear discriminant analysis
9. Reading

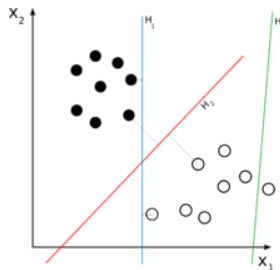
Introduction



1. In classification, the goal is to find a mapping from inputs \mathcal{X} to outputs $t \in \{1, 2, \dots, C\}$ given a labeled set of input-output pairs (**training set**)

$$S = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \dots, (\mathbf{x}_N, t_N)\}.$$

2. Each training input \mathbf{x} is a D -dimensional vector of numbers.
3. Approaches for building a classifier.
 - **Generative approach:** This approach first creates a joint model of the form of $p(\mathbf{x}, c_n)$ and then to condition on \mathbf{x} , then deriving $p(c_n|\mathbf{x})$.
 - **Discriminative approach:** This approach creates a model of the form of $p(c_n|\mathbf{x})$ directly.

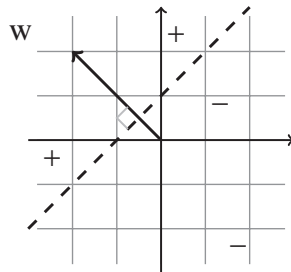


Linear classifiers



1. A linear classifier is defined as

$$g(\mathbf{x}_n) = \phi(w_1x_{n1} + w_2x_{n2} + \dots + w_Dx_{nD} + b) = \phi(\langle \mathbf{w}, \mathbf{x} \rangle + b) \in \{-1, +1\}$$



2. This classifier changes its prediction only when the argument to the sign function changes from positive to negative (or vice versa).
3. Geometrically, this transition in the feature space corresponds to crossing the **decision boundary** where the argument is exactly zero: **all \mathbf{x} such that $\mathbf{w}^T \mathbf{x} = 0$** .
4. Considering the above two dimensional data, $\mathbf{x} = (x_1, x_2)$, the hyperplane intersects the horizontal axis at $-\frac{b}{w_1}$ and the vertical axis at $-\frac{b}{w_2}$.
5. Instances that are **above** the hyperplane (share an acute angle with \mathbf{w}) are labeled **positively** and instances that are **below** the hyperplane (share an obtuse angle with \mathbf{w}) are labeled **negatively**.



1. Linear programs are problems that can be expressed as maximizing a linear function subject to linear inequalities. That is

$$\begin{aligned} & \max_{\mathbf{w} \in \mathbb{R}^D} \langle \mathbf{u}, \mathbf{w} \rangle \\ & \text{subject to } \mathbf{A}\mathbf{w} \geq \mathbf{v}. \end{aligned}$$

where

- $\mathbf{w} \in \mathbb{R}^D$ is the vector of variables we wish to determine.
 - \mathbf{A} is an $N \times D$ matrix.
 - $\mathbf{v} \in \mathbb{R}^N$ and $\mathbf{u} \in \mathbb{R}^D$ are vectors.
2. Linear programs can be solved efficiently.



1. Suppose that the training data is linearly separable.
2. We are interested to find \mathbf{w} and b that results in zero training error.
3. Let $\mathbf{w} = (b, w_1, w_2, \dots, w_D)$ and $\mathbf{x} = (1, x_1, \dots, x_D)$.
4. Hence, we are looking for $\mathbf{w} \in \mathbb{R}^{D+1}$ such that for all i

$$\text{sign}(\langle \mathbf{w}, \mathbf{x}_i \rangle) = t_i$$

5. Equivalently, we are looking for $w \in \mathbb{R}^{D+1}$ such that for all i

$$t_i \langle \mathbf{w}, \mathbf{x}_i \rangle > 0.$$

6. Let \mathbf{w}^* be a vector that satisfies this condition.
7. Define $\gamma = \min_i (t_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle)$ and let $\bar{\mathbf{w}} = \frac{\mathbf{w}^*}{\gamma}$. Therefore, for all i we have

$$t_i \langle \bar{\mathbf{w}}, \mathbf{x}_i \rangle = \frac{1}{\gamma} t_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle \geq 1.$$

8. We have thus shown that there exists a vector that for all i satisfies

$$t_i \langle \bar{\mathbf{w}}, \mathbf{x}_i \rangle > 1.$$



1. We have thus shown that there exists a vector that for all i satisfies

$$t_i \langle \bar{\mathbf{w}}, \mathbf{x}_i \rangle > 1.$$

2. To find a vector that satisfies the above inequality,
 - Set \mathbf{A} to be $N \times D$ matrix whose rows are the instances multiplied by t_i $A_{ij} = t_i \times x_{ij}$.
 - Set \mathbf{v} to be $(1, 1, \dots, 1) \in \mathbb{R}^N$.
3. Then the above inequality becomes

$$\mathbf{A}\bar{\mathbf{w}} > \mathbf{v}.$$

4. The LP form requires a maximization objective, thus, we set a **dummy** objective, $\mathbf{u} = (0, \dots, 0) \in \mathbb{R}^D$.

Perceptron algorithm



1. We would like to find a linear classifier that makes the fewest mistakes on the training set. In other words, we want find \mathbf{w} that minimizes the training error

$$\begin{aligned} E_E(\mathbf{w}) &= \frac{1}{N} \sum_{n=1}^N (1 - \delta(t_n, g(\mathbf{x}_n))) \\ &= \frac{1}{N} \sum_{n=1}^N \ell(t_n, g(\mathbf{x}_n)). \end{aligned}$$

$\delta(t, t') = 1$ if $t = t'$ and 0 otherwise. ℓ is loss function called **zero-one loss**.

2. What would be a reasonable algorithm for setting the parameters \mathbf{w} ?
3. We can just incrementally adjust the parameters so as to correct **any mistakes** that the corresponding classifier makes. Such an algorithm would seem to reduce the training error that counts the **mistakes**. A simple algorithm of this type is the **Perceptron update rule**.
4. We consider each training instances one by one, cycling through all the training instances, and adjust the parameters according to (**drive it**.)

$$\mathbf{w}' = \mathbf{w} + t_n \mathbf{x}_n \quad \text{if } t_n \neq g(\mathbf{x}_n)$$

5. In other words, the parameters (classifier) is changed only if we make a mistake. These updates tend to correct mistakes.



- The parameters (classifier) is changed only if we make a mistake. To see this,
 - When we make a mistake $\text{sign}(\mathbf{w}^\top \mathbf{x}_k) \neq t_k$.
 - The inequality $t_k \mathbf{w}^\top \mathbf{x}_k < 0$ is hold.
 - Consider \mathbf{w} after updating

$$\begin{aligned}
 t_n \mathbf{w}'^\top \mathbf{x}_n &= t_n (\mathbf{w} + t_n \mathbf{x}_n)^\top \mathbf{x}_n \\
 &= t_n \mathbf{w}^\top \mathbf{x}_n + t_n^2 \mathbf{x}_n^\top \mathbf{x}_n \\
 &= t_n \mathbf{w}^\top \mathbf{x}_n + \|\mathbf{x}_n\|^2
 \end{aligned}$$

- This means that, the value of $t_k \mathbf{w}^\top \mathbf{x}_n$ increases as a result of the update (becomes more positive).
 - If we consider the same feature vector repeatedly, then we will necessarily change the parameters such that the vector is classified correctly, i.e., the value of $t_k \mathbf{w}^\top \mathbf{x}_n$ becomes positive.
- if the training examples are possible to classify correctly with a **linear classifier**, will the Perceptron algorithm find such a classifier?
 - Yes, it does, and it will converge to such a classifier in a finite number of updates (mistakes). **To drive this result (an alternative proof), please read section 3.3 of Pattern Recognition Book by Theodoridis and Koutroumbas**



1. We considered the linearly separable case in which the following inequality holds (Let $b = 0$).

$$t_n(\mathbf{w}^*)^\top \mathbf{x}_n > 0 \quad \text{for all } n = 1, 2, \dots, N$$

\mathbf{w}^* is the weight learned by the Perceptron algorithm.

2. Now assume we want to learn a hyperplane that classifies the training set with margin of $\gamma > 0$, i.e. we have

$$t_n(\mathbf{w}^*)^\top \mathbf{x}_n > \gamma \quad \text{for all } n = 1, 2, \dots, N$$

Parameter $\gamma > 0$ is used to ensure all examples are classified correctly with a **finite margin**.

Theorem

When $\|\mathbf{x}_n\| \leq R$ for all n and some finite R , the Perceptron algorithm needs at most $\left(\frac{R}{\gamma}\right)^2 \|\mathbf{w}^*\|^2$ updates of the weight vector (\mathbf{w}).

Outline of proof.

The convergence proof is based on combining the following two results,

2.1 The inner product $(\mathbf{w}^*)^\top \mathbf{w}^{(k)}$ increases at least linearly with each update.

2.2 The squared norm $\|\mathbf{w}^{(k)}\|^2$ increases at most linearly in the number of updates k .

□



We now give details of each part.

Proof of part 1.

1. The weight vector \mathbf{w} updated when the training instance is not classified correctly. We consider the inner product $(\mathbf{w}^*)^\top \mathbf{w}^{(k)}$ before and after each update.

$$\begin{aligned}(\mathbf{w}^*)^\top \mathbf{w}^{(k)} &= (\mathbf{w}^*)^\top (\mathbf{w}^{(k-1)} + t_n \mathbf{x}_n) \\ &= (\mathbf{w}^*)^\top \mathbf{w}^{(k-1)} + t_n (\mathbf{w}^*)^\top \mathbf{x}_n \\ &\geq (\mathbf{w}^*)^\top \mathbf{w}^{(k-1)} + \gamma \\ &\geq (\mathbf{w}^*)^\top \mathbf{w}^{(k-2)} + 2\gamma \\ &\geq (\mathbf{w}^*)^\top \mathbf{w}^{(k-3)} + 3\gamma \\ &\vdots \\ &\geq (\mathbf{w}^*)^\top \mathbf{w}^{(0)} + k\gamma \\ &= k\gamma\end{aligned}$$

□



We now give details of part 2.

Proof of part 2.

1. The weight vector \mathbf{w} updated when the training instance is not classified correctly. We consider

$\|\mathbf{w}^{(k)}\|^2$ before and after each update.

$$\begin{aligned}\|\mathbf{w}^{(k)}\|^2 &= \|\mathbf{w}^{(k-1)} + t_n \mathbf{x}_n\|^2 \\ &= \|\mathbf{w}^{(k-1)}\|^2 + 2t_n (\mathbf{w}^{(k-1)})^\top \mathbf{x}_n + \|t_n \mathbf{x}_n\|^2 \\ &= \|\mathbf{w}^{(k-1)}\|^2 + 2t_n (\mathbf{w}^{(k-1)})^\top \mathbf{x}_n + \|\mathbf{x}_n\|^2 \\ &\leq \|\mathbf{w}^{(k-1)}\|^2 + \|\mathbf{x}_n\|^2 \\ &\leq \|\mathbf{w}^{(k-1)}\|^2 + R^2 \\ &\leq \|\mathbf{w}^{(k-2)}\|^2 + 2R^2 \\ &\vdots \\ &\leq \|\mathbf{w}^{(0)}\|^2 + kR^2 = kR^2\end{aligned}$$

□



We now combine two parts.

Combination of parts 1 & 2.

1. The $\cos(\mathbf{x}, \mathbf{y})$ measures the similarity of \mathbf{x} and \mathbf{y} .

$$\begin{aligned} \cos(\mathbf{w}^*, \mathbf{w}^{(k)}) &= \frac{(\mathbf{w}^*)^\top \mathbf{w}^{(k)}}{\|(\mathbf{w}^*)^\top\| \|\mathbf{w}^{(k)}\|} \\ &\stackrel{1}{\geq} \frac{k\gamma}{\|(\mathbf{w}^*)^\top\| \|\mathbf{w}^{(k)}\|} \\ &\stackrel{2}{\geq} \frac{k\gamma}{\sqrt{kR^2} \|(\mathbf{w}^*)^\top\|} \leq 1. \end{aligned}$$

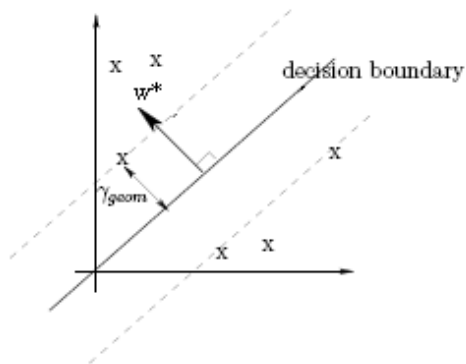
2. The last inequality is because the \cos is bounded by **one**. Hence, we have

$$\begin{aligned} k &\leq \frac{\sqrt{kR^2} \|(\mathbf{w}^*)^\top\|}{\gamma} \\ &\leq \left(\frac{R}{\gamma}\right)^2 \|(\mathbf{w}^*)^\top\|^2 = R^2 \left(\frac{\|\mathbf{w}^*\|}{\gamma}\right)^2 \end{aligned}$$

□



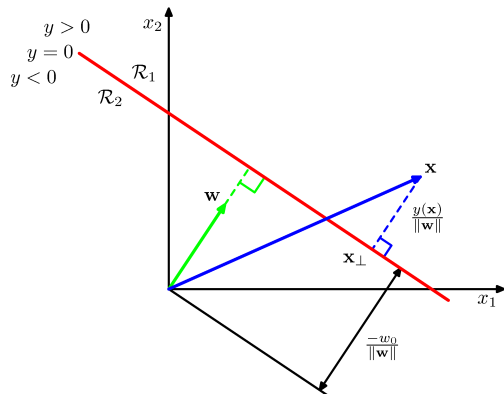
1. Does $\frac{\|\mathbf{w}^*\|}{\gamma}$ relate to the difficulty of the classification problem?
2. **Yes**, its inverse ($\frac{\gamma}{\|\mathbf{w}^*\|}$) is the smallest distance in the feature space from any example to the decision boundary specified by \mathbf{w}^* . In other words, it is a measure of separation of two classes. This distance is called **geometric distance** and denoted by γ_{geom} .



3. To calculate the geometric distance, the distance from the decision boundary to one of the examples \mathbf{x}_n for which $t_n(\mathbf{w}^*)^\top \mathbf{x}_n = \gamma$ is measured.
4. Since \mathbf{w}^* is normal to the decision boundary, the shortest path from the boundary to the instance \mathbf{x}_n will be parallel to the normal.
5. The instance for which $t_n(\mathbf{w}^*)^\top \mathbf{x}_n = \gamma$ is therefore among those closest to the boundary.



- Let \mathbf{x} be an arbitrary point and let \mathbf{x}_\perp be its orthogonal projection onto the decision surface.



- Let r be the distance between \mathbf{x} and \mathbf{x}_\perp . If $g(\mathbf{x}) = +1$, we have

$$\mathbf{x} - \mathbf{x}_\perp = r \frac{\mathbf{w}^*}{\|\mathbf{w}^*\|}$$

If $g(\mathbf{x}) = -1$, we have

$$\mathbf{x} - \mathbf{x}_\perp = -r \frac{\mathbf{w}^*}{\|\mathbf{w}^*\|}$$



1. Hence, by combining the two last equations we obtain

$$\begin{aligned} \mathbf{x} &= \mathbf{x}_\perp + r \frac{t\mathbf{w}^*}{\|\mathbf{w}^*\|} \\ \mathbf{x}_\perp &= \mathbf{x} - r \frac{t\mathbf{w}^*}{\|\mathbf{w}^*\|} \end{aligned}$$

2. It remains to find the value of r such that $(\mathbf{w}^*)^\top \mathbf{x}_\perp = 0$. This is the point where the segment hits the decision boundary. Thus, we have

$$\begin{aligned} 0 = t(\mathbf{w}^*)^\top \mathbf{x}_\perp &= t(\mathbf{w}^*)^\top \left[\mathbf{x} - r \frac{t\mathbf{w}^*}{\|\mathbf{w}^*\|} \right] \\ &= t(\mathbf{w}^*)^\top \mathbf{x} - r \frac{t^2 (\mathbf{w}^*)^\top \mathbf{w}^*}{\|\mathbf{w}^*\|} \\ &= \underbrace{t(\mathbf{w}^*)^\top \mathbf{x}}_\gamma - r \frac{\|\mathbf{w}^*\|^2}{\|\mathbf{w}^*\|} \end{aligned}$$

3. Hence, we obtain

$$\begin{aligned} 0 &= \gamma - r \|\mathbf{w}^*\| \\ r &= \frac{\gamma}{\|\mathbf{w}^*\|} \end{aligned}$$



1. Hence, the number of updates for w is equal to

$$k \leq \left(\frac{R}{\gamma_{geom}} \right)^2$$

2. Note that the result does not depend (directly) on the dimension (D) of the examples, nor the number of training examples (N). The value of γ_{geom} can be interpreted as **a measure of difficulty (or complexity) of the problem** of learning linear classifiers in this setting.
3. There are other variants of perceptron algorithm such as **normalized perceptron algorithm** and **Pocket algorithm**. **please study them**.

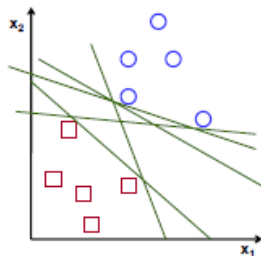


1. We have assumed that there exists a linear classifier that has a large geometric margin. This type of classifier is called **large margin classifier**.
2. We have so far used a simple iterative algorithm (**Perceptron algorithm**) to find the linear classifier.
3. Can we find a large margin classifier directly?
4. **Yes**, we can find the large margin classifier directly.
5. This classifier is known as the **support vector machine (SVM)**.

Support vector machines



1. Consider the problem of finding a separating hyperplane for a linearly separable dataset $S = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \dots, (\mathbf{x}_N, t_N)\}$ with $\mathbf{x}_i \in \mathbb{R}^D$ and $t_i \in \{-1, +1\}$.
2. Which of the infinite hyperplanes should we choose?



- Hyperplanes that pass too close to the training examples will be sensitive to noise and therefore, less likely to generalize well for data outside the training set.
 - It is reasonable to expect that a hyperplane that is farthest from all training examples will have better generalization capabilities.
3. We can find the maximum margin linear classifier by first identifying a classifier that correctly classifies all the examples and then increasing the geometric margin until we cannot increase the margin any further.
 4. We can also set up an optimization problem for directly maximizing the geometric margin.



1. We will need the classifier to be correct on all the training examples ($t_n \mathbf{w}^\top \mathbf{x}_n \geq \gamma$ for all $n = 1, 2, \dots, N$) subject to these constraints, we would like to maximize the geometric margin ($\frac{\gamma}{\|\mathbf{w}\|}$). Hence, we have (Let $\mathbf{b} = \mathbf{0}$)

$$\text{Maximize } \frac{\gamma}{\|\mathbf{w}\|} \quad \text{subject to } t_n \mathbf{w}^\top \mathbf{x}_n \geq \gamma \text{ for all } n = 1, 2, \dots, N$$

2. We can alternatively minimize the inverse $\frac{\|\mathbf{w}\|}{\gamma}$ or the inverse squared $\frac{\|\mathbf{w}\|^2}{\gamma^2}$ subject to the same constraints.

$$\text{Minimize } \frac{1}{2} \frac{\|\mathbf{w}\|^2}{\gamma^2} \quad \text{subject to } t_n \mathbf{w}^\top \mathbf{x}_n \geq \gamma \text{ for all } n = 1, 2, \dots, N$$

Factor $\frac{1}{2}$ is included merely for later convenience.

3. The above problem can be written as

$$\text{Minimize } \frac{1}{2} \left\| \frac{\mathbf{w}}{\gamma} \right\|^2 \quad \text{subject to } t_n \left(\frac{\mathbf{w}}{\gamma} \right)^\top \mathbf{x}_n \geq 1 \text{ for all } n = 1, 2, \dots, N$$

4. This problem tells the dependency on the ratio $\frac{\mathbf{w}}{\gamma}$ not \mathbf{w} or γ separately.
5. Scaling \mathbf{w} by a constant also doesn't change the decision boundary. We can therefore fix $\gamma = 1$ and solve for \mathbf{w} .



1. By fixing $\gamma = 1$ and solving for \mathbf{w} , we obtain

$$\text{Minimize } \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to } t_n \mathbf{w}^\top \mathbf{x}_n \geq 1 \text{ for all } n = 1, 2, \dots, N$$

2. This optimization problem is in the standard SVM form and is a quadratic programming problem.
3. We will modify the linear classifier here slightly by adding an offset term so that the decision boundary does not have to go through the origin. In other words, the classifier that we consider has the form

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

\mathbf{w} is the weight vector b is the bias of the separating hyperplane. The hyperplane is shown by (\mathbf{w}, b) .

4. The bias parameter changes the optimization problem to

$$\text{Minimize } \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to } t_n (\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 \text{ for all } n = 1, 2, \dots, N$$

5. Note that the bias only appears in the constraints. This is different from simply modifying the linear classifier through origin by feeding it with examples that have an additional constant component, i.e., $\mathbf{x}' = [1; \mathbf{x}]$.

Lagrangian optimization



1. Assume that we have a primal optimization problem of the form,

$$\min_{\mathbf{x}} \phi(\mathbf{x}) \quad \text{subject to} \quad g_i(\mathbf{x}) \geq 0 \quad \text{for } i = 1, 2, \dots, l$$

2. Assume that ϕ is **convex** and the constraints g_i are **linear**.
3. We can construct the **Lagrangian optimization problem** as follows,

$$\max_{\alpha} \min_{\mathbf{x}} L(\mathbf{x}, \alpha) = \max_{\alpha} \min_{\mathbf{x}} \left(\phi(\mathbf{x}) - \sum_{i=1}^l \alpha_i g_i(\mathbf{x}) \right)$$

such that

$$\alpha_i \geq 0 \quad \text{for } i = 1, 2, \dots, l$$

4. The values $\alpha_1, \dots, \alpha_l$ are called the **Lagrangian multipliers**.
5. We call \mathbf{x} the **primal variable** and α the **dual variable**.



1. We have

$$\max_{\alpha} \min_{\mathbf{x}} L(\mathbf{x}, \alpha) = \max_{\alpha} \min_{\mathbf{x}} \left(\phi(\mathbf{x}) - \sum_{i=1}^l \alpha_i g_i(\mathbf{x}) \right)$$

2. Let $\mathbf{x} = \mathbf{x}^*$ be an **optimum** then

$$\max_{\alpha} L(\mathbf{x}^*, \alpha) = \max_{\alpha} \left(\phi(\mathbf{x}^*) - \sum_{i=1}^l \alpha_i g_i(\mathbf{x}^*) \right)$$

3. Let $\alpha = \alpha^*$ be an **optimum** then

$$\min_{\mathbf{x}} L(\mathbf{x}, \alpha^*) = \min_{\mathbf{x}} \left(\phi(\mathbf{x}) - \sum_{i=1}^l \alpha_i^* g_i(\mathbf{x}) \right)$$

4. This implies that our solutions are **saddle points** on the graph of the function $L(\mathbf{x}, \alpha)$

5. An important observation is that at the saddle point the identity

$$\frac{\partial L}{\partial \mathbf{x}} = 0$$

6. Here, the point \mathbf{x}^* represents **an optimum of L with respect to \mathbf{x}** .



1. Let α^* and \mathbf{x}^* be a solution to the Lagrangian such that,

$$\max_{\alpha} \min_{\mathbf{x}} L(\mathbf{x}, \alpha) = L(\mathbf{x}^*, \alpha^*) = \phi(\mathbf{x}^*) - \sum_{i=1}^l \alpha_i^* g_i(\mathbf{x}^*)$$

2. Then \mathbf{x}^* is a solution to the primal objective function if and only if the following conditions hold

$$\frac{\partial}{\partial \mathbf{x}} L(\mathbf{x}^*, \alpha^*) = 0,$$

$$\alpha_i^* g_i(\mathbf{x}^*) = 0,$$

$$g_i(\mathbf{x}^*) \geq 0,$$

$$\alpha_i^* \geq 0,$$

for $i = 1, 2, \dots, l$.

3. These conditions are collectively referred to as the [Karush-Kuhn-Tucker \(KKT\)](#) conditions and if satisfied ensure that (Why? Please verify it.)

$$L(\mathbf{x}^*, \alpha^*) = \phi(\mathbf{x}^*)$$

4. **The KKT conditions are always satisfied for convex optimization problems.**



1. Assume that \mathbf{x}^* be an optimum, that is,

$$\frac{\partial}{\partial \mathbf{x}} L(\mathbf{x}^*, \alpha) = 0,$$

2. Then we can rewrite our Lagrangian as an objective function of only the dual variable,

$$L(\mathbf{x}^*, \alpha) = \psi(\alpha),$$

the function ψ the **Lagrangian dual**.

3. This gives us our new, **dual optimization problem**

$$\max_{\alpha} \psi(\alpha) \quad \text{subject to} \quad \alpha_i \geq 0 \quad \text{for } i = 1, 2, \dots, l$$

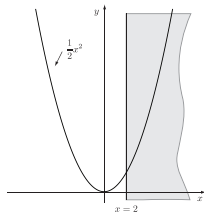
4. If the KKT conditions are satisfied

$$\max_{\alpha} \psi(\alpha) = \psi(\alpha^*) = L(\alpha^*, \mathbf{x}^*) = \phi(\mathbf{x}^*).$$



1. Consider the convex optimization problem,

$$\min_x \phi(\alpha) = \min_x \frac{1}{2}x^2 \quad \text{subject to } g(x) = x - 2 \geq 0$$



2. The Lagrangian is

$$L(x, \alpha) = \frac{1}{2}x^2 - \alpha(x - 2).$$

3. This saddle point occurs where the gradient of the Lagrangian with respect to x is equal to zero,

$$\frac{\partial L}{\partial x}(\alpha, x^*) = x^* - \alpha = 0$$

4. Solving for x^* gives $x^* = \alpha$. Now, substituting $x^* = \alpha$ into the Lagrangian gives

$$L(\alpha, x^*) = \frac{1}{2}\alpha^2 - \alpha^2 + 2\alpha = 2\alpha - \frac{1}{2}\alpha^2$$



1. We can write **dual optimization form** with $\psi(\alpha) = L(\alpha, x^*)$ as

$$\max_{\alpha} \psi(\alpha) = \max_{\alpha} \left(2\alpha - \frac{1}{2}\alpha^2 \right) \quad \text{subject to } \alpha \geq 0$$

2. Since $L(x, \alpha)$ is convex, we can write

$$\frac{\partial \psi}{\partial \alpha}(\alpha^*) = 2 - \alpha = 0$$

3. This means that $x^* = \alpha^* = 2$.

Support vector machines (cont.)



1. The optimization problem for SVM is defined as

$$\text{Minimize } \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to } t_n (\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 \text{ for all } n = 1, 2, \dots, N$$

2. In order to solve this constrained optimization problem, we introduce Lagrange multipliers $\alpha_n \geq 0$, with one multiplier α_n for each of the constraints giving the Lagrangian function

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{n=1}^N \alpha_n \left[t_n (\mathbf{w}^\top \mathbf{x}_n + b) - 1 \right]$$

where $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_N)^\top$.

3. Note the minus sign in front of the Lagrange multiplier term, because we are minimizing with respect to \mathbf{w} and b , and maximizing with respect to α . **please read Appendix E of Bishop.**
4. Setting the derivatives of $L(\mathbf{w}, b, \alpha)$ with respect to \mathbf{w} and b equal to zero, we obtain the following two equations

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

$$\mathbf{w} = \sum_{n=1}^N \alpha_n t_n \mathbf{x}_n$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow$$

$$0 = \sum_{n=1}^N \alpha_n t_n$$



1. L has to be minimized with respect to the primal variables \mathbf{w} and b and maximized with respect to the dual variables α_n . Eliminating \mathbf{w} and b from $L(\mathbf{w}, b, a)$ using these conditions then gives the dual representation of the problem in which we maximize

$$\psi(\alpha) = \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_n \alpha_m t_n t_m \mathbf{x}_n^\top \mathbf{x}_m$$

2. We need to maximize $\psi(\alpha)$ subject to the following constraints

$$\begin{aligned} \alpha_n &\geq 0 \quad \forall n \\ \sum_{n=1}^N \alpha_n t_n &= 0 \end{aligned}$$

3. The constrained optimization of this form satisfies the Karush-Kuhn-Tucker (KKT) conditions, which in this case require that the following three properties hold

$$\begin{aligned} \alpha_n &\geq 0 \\ t_n g(\mathbf{x}_n) &\geq 1 \\ \alpha_n [t_n g(\mathbf{x}_n) - 1] &= 0 \end{aligned}$$



1. For optimal α_n 's,

$$\alpha_n \left[1 - t_n (\mathbf{w}^\top \mathbf{x}_n + b) \right] = 0$$

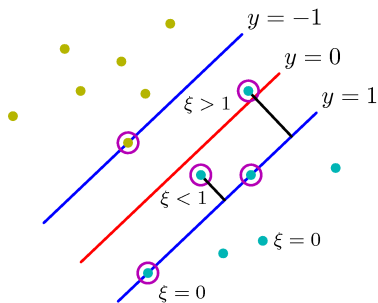
2. α_n is **non-zero** only if \mathbf{x}_n lies on one of the **two margin boundaries**, i.e., for which $t_n(\mathbf{w}^\top \mathbf{x}_n + b) = 1$
3. These examples are called **support vectors**.
4. To classify a data \mathbf{x} using the trained model, we evaluate the sign of $g(\mathbf{x})$ defined by

$$g(\mathbf{x}) = \sum_{n=1}^N \alpha_n t_n \mathbf{x}_n^\top \mathbf{x}$$

5. How do you find b ?



1. We have assumed that the training data are linearly separable in the feature space. The resulting SVM will give exact separation of the training data.
2. In the practice, the class-conditional distributions may overlap, in which the exact separation of the training data can lead to poor generalization.
3. We need a way to modify the SVM so as to allow some training examples to be miss-classified.
4. To do this, we introduce **slack variables** ($\xi_n \geq 0$) (**distance by which it violates the margin**); one slack variable for each training example.
5. The slack variables are defined by $\xi_n = 0$ for examples that are inside the correct boundary margin and $\xi_n = |t_n - g(x_n)|$ for other examples.
6. Thus for data point that is on the decision boundary $g(x_n) = 0$ will have $\xi_n = 1$ and the data points with $\xi_n \geq 1$ will be misclassified.





1. The exact classification constraints will be

$$t_n g(x_n) \geq 1 - \xi_n \quad \text{for } n = 1, 2, \dots, N$$

2. Our goal is now to maximize the margin while softly penalizing points that lie on the wrong side of the margin boundary. We therefore minimize

$$C \sum_{n=1}^N \xi_n + \frac{1}{2} \|w\|^2$$

$C > 0$ controls the trade-off between the slack variable penalty and the margin.

3. We now wish to solve the following optimization problem.

$$\text{Minimize } \frac{1}{2} \|w\|^2 + C \sum_{n=1}^N \xi_n \quad \text{subject to } t_n g(x_n) \geq 1 - \xi_n \text{ for all } n = 1, 2, \dots, N$$

4. The corresponding Lagrangian is given

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 + C \sum_{n=1}^N \xi_n - \sum_{n=1}^N \alpha_n [t_n g(x_n) - 1 + \xi_n] - \sum_{n=1}^N \beta_n \xi_n$$

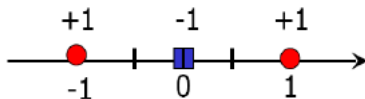
where $\alpha_n \geq 0$ and $\beta_n \geq 0$ are Lagrange multipliers.

5. please read [section 7.1.1 of Bishop](#).

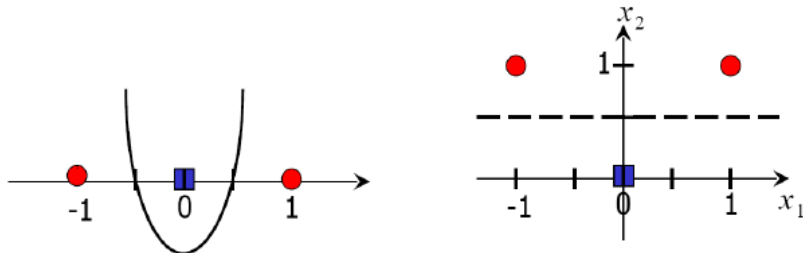
Non-linear support vector machine



1. Most data sets are not linearly separable, for example



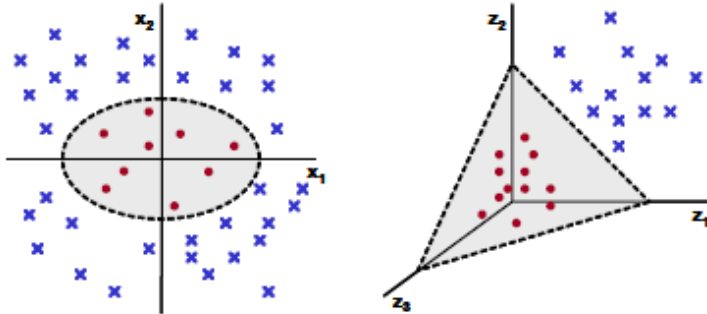
2. Instances that are not linearly separable in 1–dimension, may be linearly separable in 2–dimensions, for example



3. In this case, we have two solutions
 - Increase dimensionality of data set by introducing mapping ϕ .
 - Use a more complex model for classifier.



1. To solve the non-linearly separable dataset, we use mapping ϕ .
2. For example, let $x = (x_1, x_2)^\top$, $z = (z_1, z_2, z_3)^\top$, and $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$. If we use mapping $z = \phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^\top$, the dataset will be linearly separable in \mathbb{R}^3 .



3. Mapping dataset to higher dimensions has two major problems.
 - In high dimensions, there is risk of over-fitting.
 - In high dimensions, we have more computational cost.
4. The generalization capability in higher dimension is ensured by using large margin classifiers.
5. The mapping is an implicit mapping not explicit.



1. The SVM uses the following discriminant function.

$$g(x) = \sum_{n=1}^N \alpha_n t_n x_n^\top x$$

2. This solution depends on the dot-product between two points x_n and x .
3. The operation in high dimensional space $\phi(x)$ don't have performed explicitly if we can find a function $K(x_i, x_j)$ such that $K(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$.
4. $K(x_i, x_j)$ is called **kernel** in the SVM.
5. Suppose $x, z \in \mathbb{R}^D$ and consider the following kernel

$$K(x, z) = (x^\top z)^2$$

6. It is a valid kernel because

$$\begin{aligned} K(x, z) &= \left(\sum_{i=1}^D x_i z_i \right) \left(\sum_{j=1}^D x_j z_j \right) \\ &= \sum_{i=1}^D \sum_{j=1}^D (x_i x_j) (z_i z_j) = \phi(x)^\top \phi(z) \end{aligned}$$

where the mapping ϕ for $D = 2$ is

$$\phi(x) = (x_1 x_1, x_1 x_2, x_2 x_1, x_2 x_2)^\top$$



1. Show that kernel $K(x, z) = (x^\top z + c)^2$ is a valid kernel.
2. A kernel K is valid if there is some mapping ϕ such that $K(x, z) = \phi(x)^\top \phi(z)$.
3. Assume that K is a valid kernel. Consider a set of N points, K is $N \times N$ square matrix defined as

$$K = \begin{pmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_N) \\ k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_N, x_1) & k(x_N, x_2) & \cdots & k(x_N, x_N) \end{pmatrix}$$

K is called **kernel matrix**.

4. If K is a valid kernel then

$$k_{ij} = k(x_i, x_j) = \phi(x_i)^\top \phi(x_j) = \phi(x_j)^\top \phi(x_i) = k(x_j, x_i) = k_{ji}$$

5. Thus K is symmetric. It can also be shown that K is positive semi-definite (**show it**).
6. Thus if K is a **valid kernel**, then the corresponding kernel matrix is **symmetric positive semi-definite**. It is both **necessary** and **sufficient** conditions for K to be a valid kernel.

**Theorem (Mercer)**

Assume that $K : \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}$. Then for K to be a valid (Mercer) kernel, it is necessary and sufficient that for any $\{x_1, x_2, \dots, x_N\}$, ($N > 1$) the corresponding kernel matrix is **symmetric positive semi-definite**.

1. Some valid kernel functions

- Polynomial kernels

$$K(x, z) = (x^\top z + 1)^p$$

p is the degree of the polynomial and specified by the user.

- Radial basis function kernels

$$K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$$

The width σ is specified by the user. This kernel corresponds to an infinite dimensional mapping ϕ .

- Sigmoid Kernel

$$K(x, z) = \tanh\left(\beta_0 x^\top z + \beta_1\right)$$

This kernel only meets Mercer's condition for certain values of β_0 and β_1 .



1. Advantages

- The problem doesn't have local minima and we can find its optimal solution in polynomial time.
- The solution is stable, repeatable, and sparse (it only involves the support vectors).
- The user must select a few parameters such as the penalty term C and the kernel function and its parameters.
- The algorithm provides a method to control complexity independently of dimensionality.
- SVMs have been shown (theoretically and empirically) to have excellent generalization capabilities.

2. Disadvantages

- There is no method for choosing the kernel function and its parameters.
- It is not a straight forward method to extend SVM to multi-class classifiers.
- Predictions from a SVM are not probabilistic.
- It has high algorithmic complexity and needs extensive memory to be used in large-scale tasks.

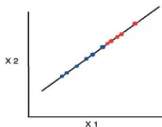
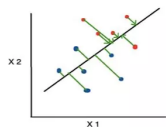
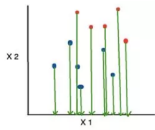
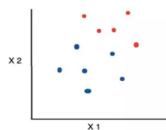
Linear discriminant analysis



1. One way to view a linear classification model is in terms of **dimensionality reduction**.
2. Consider a two-class problem and suppose we take a D -dimensional input vector x and project it down to one dimension using

$$z = W^T x$$

3. If we place a threshold on z and classify $z \geq -w_0$ as class C_1 , and otherwise class C_2 , then we obtain our standard linear classifier.
4. In general, the projection onto one dimension leads to a considerable loss of information, and classes that are well separated in the original D -dimensional space may become strongly overlapping in one dimension.





1. However, by adjusting the components of the weight vector W , we can select a projection that maximizes the class separation.
2. Consider a two-class problem in which there are N_1 points of class C_1 and N_2 points of class C_2 . Hence the mean vectors of the class C_j is given by

$$\mathbf{m}_j = \frac{1}{N_j} \sum_{i \in C_j} x_i$$

3. The simplest measure of the separation of the classes, when projected onto W , is the **separation of the projected class means**.
4. This suggests that we might choose W so as to maximize

$$m_2 - m_1 = W^T (\mathbf{m}_2 - \mathbf{m}_1)$$

where

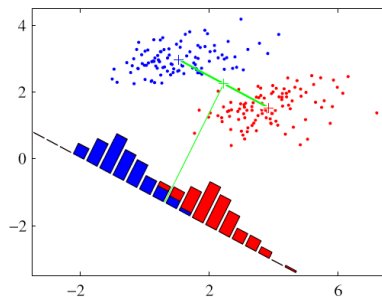
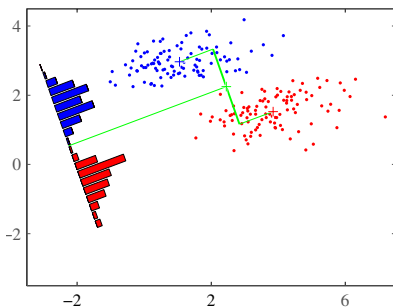
$$m_j = W^T \mathbf{m}_j$$

5. This expression can be made arbitrarily large simply by increasing the magnitude of W .
6. To solve this problem, we could constrain W to have unit length, so that $\sum_i w_i^2 = 1$.
7. Using a Lagrange multiplier to perform the constrained maximization, we then find that

$$W \propto (\mathbf{m}_2 - \mathbf{m}_1)$$



1. This approach has a problem: The following figure shows two classes that are well separated in the original two dimensional space but that have considerable overlap when projected onto the line joining their means.



2. This difficulty arises from the strongly **non-diagonal covariances** of the class distributions.
3. The idea proposed by Fisher is to **maximize a function** that will give a **large separation between the projected class means** while also giving a **small variance within each class**, thereby **minimizing the class overlap**.



1. The idea proposed by Fisher is to **maximize a function** that will give a **large separation between the projected class means** while also giving a **small variance within each class**, thereby **minimizing the class overlap**.
2. The projection $\mathbf{z} = \mathbf{W}^T \mathbf{x}$ transforms the set of labeled data points in \mathbf{x} into a labeled set in the one-dimensional space z .
3. The within-class variance of the transformed data from class C_j equals

$$s_j^2 = \sum_{i \in C_j} (z_i - m_j)^2$$

where

$$z_i = \mathbf{w}^T \mathbf{x}_i.$$

4. We can define the total within-class variance for the whole data set to be $s_1^2 + s_2^2$.
5. The Fisher criterion is defined to be the ratio of the between-class variance to the within-class variance and is given by

$$J(\mathbf{W}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$



1. Between-class covariance matrix equals to

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^\top$$

2. Total within-class covariance matrix equals to

$$\mathbf{S}_W = \sum_{i \in C_1} (x_i - \mathbf{m}_1)(x_i - \mathbf{m}_1)^\top + \sum_{i \in C_2} (x_i - \mathbf{m}_2)(x_i - \mathbf{m}_2)^\top$$

3. We have

$$\begin{aligned} (m_1 - m_2)^2 &= (W^\top \mathbf{m}_1 - W^\top \mathbf{m}_2)^2 \\ &= W^\top \underbrace{(\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^\top}_{S_B} W \\ &= W^\top S_B W, \end{aligned}$$



1. Also we have

$$\begin{aligned}
 s_1^2 &= \sum_i \left(W^\top x_i - \mathbf{m}_1 \right)^2 \\
 &= \sum_i W^\top (x_i - \mathbf{m}_1) (x_i - \mathbf{m}_1)^\top W \\
 &= W^\top \underbrace{\left[\sum_i ((x_i - \mathbf{m}_1)(x_i - \mathbf{m}_1)^\top) \right]}_{S_1} W \\
 &= W^\top S_1 W,
 \end{aligned}$$

2. and $S_W = S_1 + S_2$. Hence, $J(W)$ can be written as

$$J(w) = \frac{W^\top S_B W}{W^\top S_W W}$$

3. Derivative of $J(W)$ with respect to W equals to (using $\frac{\partial x^\top A x}{\partial x} = (A + A^\top)x$)

$$\frac{\partial J(W)}{\partial W} = \frac{[S_B + S_B^\top] W W^\top S_W W - [S_W + S_W^\top] W W^\top S_B W}{(W^\top S_W W)^2}$$



1. Since \mathbf{S}_B and \mathbf{S}_W are symmetric, we obtain

$$\frac{\partial J(W)}{\partial W} = \frac{[\mathbf{S}_B + \mathbf{S}_B^T] WW^T \mathbf{S}_W W - [\mathbf{S}_W + \mathbf{S}_W^T] WW^T \mathbf{S}_B W}{(W^T \mathbf{S}_W W)^2} = 0$$

$$\Rightarrow [W^T \mathbf{S}_W W] \mathbf{S}_B W = [W^T \mathbf{S}_B W] \mathbf{S}_W W$$

2. Dividing by $W^T \mathbf{S}_W W$

$$\left[\frac{W^T \mathbf{S}_W W}{W^T \mathbf{S}_W W} \right] \mathbf{S}_B W = \left[\frac{W^T \mathbf{S}_B W}{W^T \mathbf{S}_W W} \right] \mathbf{S}_W W$$

$$\Rightarrow \mathbf{S}_B W = \lambda \mathbf{S}_W W$$

3. If \mathbf{S}_W is invertible, we have

$$\mathbf{S}_W^{-1} \mathbf{S}_B W = \lambda W$$

4. Solving the above generalized eigenvalue problem yields

$$W \propto \mathbf{S}_W^{-1} (\mathbf{m}_2 - \mathbf{m}_1)$$






1. The result $W \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$ is known as **Fisher's linear discriminant**.
2. Although strictly it is not a discriminant but rather a specific choice of direction for projection of the data down to one dimension.
3. The above idea can be extended to multiple classes (**Read section 4.1.6 of Bishop**).

Reading



1. Sections 4.1 & 7.1 of [Pattern Recognition and Machine Learning Book](#) (Bishop 2006).
2. Sections 8.5.4, 14.4, & 14.5 of [Machine Learning: A probabilistic perspective](#) (Murphy 2012).
3. Sections 10.2.5, 17.1, 17.2, & 17.5 of [Probabilistic Machine Learning: An introduction](#) (Murphy 2022).



-  Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag.
-  Murphy, Kevin P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.
-  — (2022). *Probabilistic Machine Learning: An introduction*. The MIT Press.

Questions?