Linear & nonlinear classifiers
Machine Learning

Hamid Beigy
Sharif University of Technology

Fall 1396
Table of contents

1 Introduction
2 Linear classifiers through origin
3 Perceptron algorithm
4 Support vector machines
5 Lagrangian optimization
6 Support vector machines (cont.)
7 Non-linear support vector machine
8 Generalized linear classifier
9 Linear discriminant analysis
Table of contents

1. Introduction
2. Linear classifiers through origin
3. Perceptron algorithm
4. Support vector machines
5. Lagrangian optimization
6. Support vector machines (cont.)
7. Non-linear support vector machine
8. Generalized linear classifier
9. Linear discriminant analysis
In classification, the goal is to find a mapping from inputs $X$ to outputs $t \in \{1, 2, \ldots, C\}$ given a labeled set of input-output pairs (training set)

$$S = \{(x_1, t_1), (x_2, t_2), \ldots, (x_N, t_N)\}.$$

Each training input $x$ is a $D$–dimensional vector of numbers.

Approaches for building a classifier.
- **Generative approach**: This approach first creates a joint model of the form of $p(x, C_n)$ and then to condition on $x$, then deriving $p(C_n|x)$.
- **Discriminative approach**: This approach creates a model of the form of $p(C_n|x)$ directly.
Table of contents

1 Introduction

2 Linear classifiers through origin

3 Perceptron algorithm

4 Support vector machines

5 Lagrangian optimization

6 Support vector machines (cont.)

7 Non-linear support vector machine

8 Generalized linear classifier

9 Linear discriminant analysis
We consider the following type of linear classifiers.

\[ y(x_n) = g(x_n) = \text{sign}(w_1x_{n1} + w_2x_{n2} + \ldots + w_Dx_{nD}) \in \{-1, +1\} \]

\[ = \text{sign}\left(\sum_{j=1}^{D} w_n x_{nj}\right) = \text{sign}\left(w^T x_n\right). \]

- \( w = (w_1, w_2, \ldots, w_D)^T \) is a column vector of real valued parameters (\( w \in \mathbb{R}^D \)). Different values of \( w \) give different functions.
- \( x_n = (x_{n1}, x_{n2}, \ldots, x_{nD})^T \) is a column vector of real values.
- This classifier changes its prediction only when the argument to the sign function changes from positive to negative (or vice versa).
- Geometrically, this transition in the feature space corresponds to crossing the decision boundary where the argument is exactly zero: all \( x \) such that \( w^T x = 0 \).
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Linear classifiers through origin</td>
</tr>
<tr>
<td>3</td>
<td>Perceptron algorithm</td>
</tr>
<tr>
<td>4</td>
<td>Support vector machines</td>
</tr>
<tr>
<td>5</td>
<td>Lagrangian optimization</td>
</tr>
<tr>
<td>6</td>
<td>Support vector machines (cont.)</td>
</tr>
<tr>
<td>7</td>
<td>Non-linear support vector machine</td>
</tr>
<tr>
<td>8</td>
<td>Generalized linear classifier</td>
</tr>
<tr>
<td>9</td>
<td>Linear discriminant analysis</td>
</tr>
</tbody>
</table>
The Perceptron algorithm

- We would like to find a linear classifier that makes the fewest mistakes on the training set. In other words, we want to find $w$ that minimizes the training error

$$E_E(w) = \frac{1}{N} \sum_{n=1}^{N} (1 - \delta(t_n, g(x_n)))$$

$$= \frac{1}{N} \sum_{n=1}^{N} \ell(t_n, g(x_n)).$$

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

- What would be a reasonable algorithm for setting the parameters $w$?

- 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. The simplest algorithm of this type is the Perceptron update rule.

- We consider each training instance one by one, cycling through all the training instances, and adjust the parameters according to (drive it.)

$$w' = w + t_n x_n \text{ if } t_n \neq g(x_n)$$

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

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

\[
\begin{align*}
t_n w' T x &= t_n (w + t_n x_n)^T x_n \\
&= t_n w^T x_n + t_n^2 x_n^T x_n \\
&= t_n w^T x_n + ||x_n||^2
\end{align*}
\]

- This means that, the value of \( t_k w^T x_n \) 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 \( t_k w^T x_n \) 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.
The Perceptron algorithm (cont.)

- We considered the linearly separable case in which the following inequality holds.
  \[ t_n(w^*)^T x_n > 0 \quad \text{for all } n = 1, 2, \ldots, N \]
  
  $W^*$ is the weight learned by the Perceptron algorithm.

- Now assume we want to learn a hyperplane that classifies the training set with margin of $\gamma > 0$, i.e. we have
  \[ t_n(w^*)^T x_n > \gamma \quad \text{for all } n = 1, 2, \ldots, N \]

  Parameter $\gamma > 0$ is used to ensure that each example is classified correctly with a finite margin.

**Theorem**

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

**Outline of proof.**

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

1. The inner product $(w^*)^T w^{(k)}$ increases at least linearly with each update.
2. The squared norm $||w^{(k)}||^2$ increases at most linearly in the number of updates $k$. 
We now give details of each part.

**Proof of part 1.**

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

\[
(w^*)^T w^{(k)} = (w^*)^T \left( w^{(k-1)} + t_n x_n \right)
\]
\[
= (w^*)^T w^{(k-1)} + t_n (w^*)^T x_n
\]
\[
\geq (w^*)^T w^{(k-1)} + \gamma
\]
\[
\geq (w^*)^T w^{(k-2)} + 2\gamma
\]
\[
\geq (w^*)^T w^{(k-3)} + 3\gamma
\]
\[
\vdots
\]
\[
\geq (w^*)^T w^{(0)} + k\gamma
\]
\[
= k\gamma
\]
The Perceptron algorithm (cont.)

- We now give details of part 2.

**Proof of part 2.**

The weight vector $w$ updated when the training instance is not classified correctly. We consider $\|w(k)\|^2$ before and after each update.

\[
\|w(k)\|^2 = \|w(k-1) + t_n x_n\|^2 \\
= \|w(k-1)\|^2 + 2t_n (w(k-1))^T x_n + \|t_n x_n\|^2 \\
= \|w(k-1)\|^2 + 2t_n (w(k-1))^T x_n + \|x_n\|^2 \\
\leq \|w(k-1)\|^2 + \|x_n\|^2 \\
\leq \|w(k-1)\|^2 + R^2 \\
\leq \|w(k-2)\|^2 + 2R^2 \\
\vdots \\
\leq \|w(0)\|^2 + kR^2 = kR^2
\]
The Perceptron algorithm (cont.)

- We now combine two parts.

**Combination of parts 1 & 2.**

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

   $$\cos\left(w^*, w^{(k)}\right) = \frac{(w^*)^T w^{(k)}}{\|w^*\|^T \|w^{(k)}\|}$$

   $$\leq \frac{1}{k\gamma} \left\|w^*\right\|$$

   $$\leq \frac{2}{\sqrt{kR^2\|w^*\|^T}}$$

   $$\leq 1$$

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

   $$k \leq \frac{\sqrt{kR^2\|w^*\|^T}}{\gamma}$$

   $$\leq \left(\frac{R}{\gamma}\right)^2 \|w^*\|^T = R^2 \left(\frac{\|w^*\|}{\gamma}\right)^2$$
Does $\|w^*\|$ relate to the difficulty of the classification problem?

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

In order to calculate the geometric distance ($\gamma_{geom}$), the distance from the decision boundary ($\langle w^* \rangle^T x = 0$) to one of the examples $x_n$ for which $t_n(w^*)^T x_n = \gamma$ is measured. Since $w^*$ is normal to the decision boundary, the shortest path from the boundary to the instance $x_n$ will be parallel to the normal. The instance for which $t_n(w^*)^T x_n = \gamma$ is therefore among those closest to the boundary.
Let $x$ be an arbitrary point and let $x_\perp$ be its orthogonal projection onto the decision surface.

Let $r$ be the distance between $x$ and $x_\perp$. If $g(x) = +1$, we have

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

If $g(x) = -1$, we have

$$x - x_\perp = -r \frac{w^*}{\|w^*\|}$$
Hence, by combining the two last equations we obtain
\[
x = x_\perp + r \frac{tw^*}{\|w^*\|}
\]
\[
x_\perp = x - r \frac{tw^*}{\|w^*\|}
\]

It remains to find the value of \( r \) such that \((w^*)^T x_\perp = 0\). This is the point where the segment hits the decision boundary. Thus, we have
\[
0 = t(w^*)^T x_\perp = t(w^*)^T \left[ x - r \frac{tw^*}{\|w^*\|} \right]
\]
\[
= t(w^*)^T x - r \frac{t^2(w^*)^T w^*}{\|w^*\|}
\]
\[
= t(w^*)^T x - r \frac{\|w^*\|^2}{\|w^*\|}
\]

Hence, we obtain
\[
0 = \gamma - r \|w^*\|
\]
\[
r = \frac{\gamma}{\|w^*\|}
\]
Hence, the number of updates for $w$ is equal to

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

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 $\gamma_{\text{geom}}$ can be interpreted as a measure of difficulty (or complexity) of the problem of learning linear classifiers in this setting.

There are other variants of perceptron algorithm such as normalized perceptron algorithm and Pocket algorithm. Please study them.

We have assumed that there exists a linear classifier that has a large geometric margin. This type of classifier is called large margin classifier.

We have so far used a simple iterative algorithm (perceptron algorithm) to find the linear classifier.

Can we find a large margin classifier directly?

Yes, we can find the large margin classifier directly. This classifier is known as the support vector machine (SVM).
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Linear classifiers through origin</td>
</tr>
<tr>
<td>3</td>
<td>Perceptron algorithm</td>
</tr>
<tr>
<td>4</td>
<td>Support vector machines</td>
</tr>
<tr>
<td>5</td>
<td>Lagrangian optimization</td>
</tr>
<tr>
<td>6</td>
<td>Support vector machines (cont.)</td>
</tr>
<tr>
<td>7</td>
<td>Non-linear support vector machine</td>
</tr>
<tr>
<td>8</td>
<td>Generalized linear classifier</td>
</tr>
<tr>
<td>9</td>
<td>Linear discriminant analysis</td>
</tr>
</tbody>
</table>
Support vector machines

- Consider the problem of finding a separating hyperplane for a linearly separable dataset 
  \[ S = \{(x_1, t_1), (x_2, t_2), \ldots, (x_N, t_N)\} \]
  with \( x_i \in \mathbb{R}^D \) and \( t_i \in \{-1, +1\} \).
- 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.

- 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.
- We can also set up an optimization problem for directly maximizing the geometric margin.
We will need the classifier to be correct on all the training examples \((t_nw^Tx_n \geq \gamma \text{ for all } n = 1, 2, \ldots, N)\) subject to these constraints, we would like to maximize the geometric margin \((\frac{\gamma}{\|w\|})\). Hence, we have

\[
\text{Maximize} \frac{\gamma}{\|w\|} \quad \text{subject to } t_nw^Tx_n \geq \gamma \text{ for all } n = 1, 2, \ldots, N
\]

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

\[
\text{Minimize} \frac{1}{2} \frac{\|w\|^2}{\gamma^2} \quad \text{subject to } t_nw^Tx_n \geq \gamma \text{ for all } n = 1, 2, \ldots, N
\]

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

The above problem can be written as

\[
\text{Minimize} \frac{1}{2} \left(\frac{w}{\gamma}\right)^2 \quad \text{subject to } t_n \left(\frac{w}{\gamma}\right)^T x_n \geq 1 \text{ for all } n = 1, 2, \ldots, N
\]

This problem tells the dependency on the ration \(\frac{w}{\gamma}\) not \(w\) or \(\gamma\) separately.

Scaling \(w\) by a constant also doesn’t change the decision boundary. We can therefore fix \(\gamma = 1\) and solve for \(w\).
By fixing $\gamma = 1$ and solving for $w$, we obtain

$$
\text{Minimize } \frac{1}{2} ||w||^2 \quad \text{subject to } t_n w^T x_n \geq 1 \text{ for all } n = 1, 2, \ldots, N
$$

This optimization problem is in the standard SVM form and is a quadratic programming problem.

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

$$
g(x) = w^T x + b
$$

$w$ is the weight vector $b$ is the bias of the separating hyperplane. The hyperplane is shown by $(w, b)$.

The bias parameter changes the optimization problem to

$$
\text{Minimize } \frac{1}{2} ||w||^2 \quad \text{subject to } t_n (w^T x_n + b) \geq 1 \text{ for all } n = 1, 2, \ldots, N
$$

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., $x' = [x; 1]$. 
# Table of contents

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

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

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

Assume that $\phi$ is convex the constraints $g_i$ are linear.

We can construct the Lagrangian optimization problem as follows,

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

such that

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

The values $\alpha_1, \ldots, \alpha_l$ are called the Lagrangian multipliers.

We call $x$ the primal variable and $\alpha$ the dual variable.
Lagrangian optimization (cont.)

- we have

\[
\max_\alpha \min_x L(x, \alpha) = \max_\alpha \min_x \left( \phi(x) - \sum_{i=1}^{l} \alpha_i g_i(x) \right)
\]

- Let \( x = x^* \) be an optimum then

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

- Let \( \alpha = \alpha^* \) be an optimum then

\[
\min_x L(x, \alpha^*) = \min_x \left( \phi(x) - \sum_{i=1}^{l} \alpha_i^* g_i(x) \right)
\]

- This implies that our solutions are saddle points on the graph of the function \( L(x, \alpha) \)

- An important observation is that at the saddle point the identity

\[
\frac{\partial L}{\partial x} = 0
\]

- Here, the point \( x^* \) represents an optimum of \( L \) with respect to \( x \).
Lagrangian optimization (cont.)

- Let \( \alpha^* \) and \( x^* \) be a solution to the Lagrangian such that,

\[
\max_{\alpha} \min_x L(x, \alpha) = L(x^*, \alpha^*) = \phi(x^*) - \sum_{i=1}^{l} \alpha_i^* g_i(x^*)
\]

- Then \( x^* \) is a solution to the primal objective function if and only if the following conditions hold

\[
\frac{\partial}{\partial x} L(x^*, \alpha^*) = 0, \\
\alpha_i^* g_i(x^*) = 0, \\
g_i(x^*) \geq 0, \\
\alpha_i^* \geq 0,
\]

for \( i = 1, 2, \ldots, l \).

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

\[
L(x^*, \alpha^*) = \phi(x^*)
\]

- The KKT conditions are always satisfied for convex optimization problems.
Assume that \(x^*\) be an optimum, that is,
\[
\frac{\partial}{\partial x} L(x^*, \alpha) = 0,
\]

Then we can rewrite our Lagrangian as an objective function of only the dual variable,
\[
L(x^*, \alpha) = \psi(\alpha),
\]
the function \(\psi\) the Lagrangian dual.

This gives us our new, dual optimization problem
\[
\max_{\alpha} \psi(\alpha) \quad \text{subject to} \quad \alpha_i \geq 0 \quad \text{for} \ i = 1, 2, \ldots, l
\]

If the KKT conditions are satisfied
\[
\max_{\alpha} \psi(\alpha) = \psi(\alpha^*) = L(\alpha^*, x^*) = \phi(x^*).
\]
Consider the convex optimization problem,

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

The Lagrangian is

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

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

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

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

$$L(\alpha, x^*) = \frac{1}{2} \alpha^2 - \alpha^2 + 2\alpha = 2\alpha - \frac{1}{2} \alpha^2$$
We can write dual optimization form with $\psi(\alpha) = L(\alpha, x^*)$ as

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

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

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

This means that $x^* = \alpha^* = 2$. 
The optimization problem for SVM is defined as

\[
\text{Minimize} \frac{1}{2} \|w\|^2 \quad \text{subject to} \quad t_n (w^T x_n + b) \geq 1 \quad \text{for all} \quad n = 1, 2, \ldots, N
\]

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

\[
L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} \alpha_n \left[ t_n \left( w^T x_n + b \right) - 1 \right]
\]

where \(\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_N)^T\).

Note the minus sign in front of the Lagrange multiplier term, because we are minimizing with respect to \(w\) and \(b\), and maximizing with respect to \(\alpha\). Please read Appendix E of Bishop.

Setting the derivatives of \(L(w, b, \alpha)\) with respect to \(w\) and \(b\) equal to zero, we obtain the following two equations

\[
\frac{\partial L}{\partial w} = 0 \implies w = \sum_{n=1}^{N} \alpha_n t_n x_n
\]

\[
\frac{\partial L}{\partial b} = 0 \implies 0 = \sum_{n=1}^{N} \alpha_n t_n
\]
Support vector machines (cont.)

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

\[
\tilde{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m t_n t_m x_n^T x_m
\]

- We need to maximize \( \tilde{L}(\alpha) \) subject to the following constraints

\[
\alpha_n \geq 0 \quad \forall n
\]

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

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

\[
\alpha_n \geq 0
\]

\[
t_n g(x_n) \geq 1
\]

\[
\alpha_n [t_n g(x_n) - 1] = 0
\]

- To classify a data \( x \) using the trained model, we evaluate the sign of \( g(x) \) defined by

\[
g(x) = \sum_{n=1}^{N} \alpha_n t_n x_n^T x
\]
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.

In the practice, the class-conditional distributions may overlap, in which the exact separation of the training data can lead to poor generalization.

We need a way to modify the SVM so as to allow some training examples to be miss-classified.

To do this, we introduce slack variables \((\xi_n \geq 0)\); one slack variable for each training example.

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.

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.
The exact classification constraints will be
\[ t_n g(x_n) \geq 1 - \xi_n \quad \text{for } n = 1, 2, \ldots, N \]

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.

We now wish to solve the following optimization problem.

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

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.

please read section 7.1.1 of Bishop.
<table>
<thead>
<tr>
<th></th>
<th>Table of contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Linear classifiers through origin</td>
</tr>
<tr>
<td>3</td>
<td>Perceptron algorithm</td>
</tr>
<tr>
<td>4</td>
<td>Support vector machines</td>
</tr>
<tr>
<td>5</td>
<td>Lagrangian optimization</td>
</tr>
<tr>
<td>6</td>
<td>Support vector machines (cont.)</td>
</tr>
<tr>
<td>7</td>
<td><strong>Non-linear support vector machine</strong></td>
</tr>
<tr>
<td>8</td>
<td>Generalized linear classifier</td>
</tr>
<tr>
<td>9</td>
<td>Linear discriminant analysis</td>
</tr>
</tbody>
</table>
Most data sets are not linearly separable, for example:

\[ +1 \quad -1 \quad +1 \]
\[ -1 \quad 0 \quad 1 \]

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

\[ x_1 \]
\[ x_2 \]

In this case, we have two solutions:

- Increase dimensionality of data set by introducing mapping \( \phi \).
- Use a more complex model for classifier.
Non-linear support vector machine (cont.)

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

- 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.

- The generalization capability in higher dimension is ensured by using large margin classifiers.
- The mapping is an implicit mapping not explicit.
The SVM uses the following discriminant function.

\[ g(x) = \sum_{n=1}^{N} \alpha_n t_n x_n^T x \]

This solution depends on the dot-product between two points \( x_n \) and \( x \).

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 \( K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \).

\( K(x_i, x_j) \) is called kernel in the SVM.

Suppose \( x, z \in \mathbb{R}^D \) and consider the following kernel

\[ K(x, z) = \left( x^T z \right)^2 \]

It is a valid kernel because

\[
K(x, z) = \left( \sum_{i=1}^{D} x_i z_i \right) \left( \sum_{j=1}^{D} x_j z_j \right) \\
= \sum_{i=1}^{D} \sum_{j=1}^{D} (x_i x_j) (z_i z_j) = \phi(x)^T \phi(z)
\]

where the mapping \( \phi \) for \( D = 2 \) is

\[ \phi(x) = (x_1 x_1, x_1 x_2, x_2 x_1, x_2 x_2)^T \]
Non-linear support vector machine (cont.)

- Show that kernel \( K(x, z) = (x^T z + c)^2 \) is a valid kernel.
- A kernel \( K \) is valid if there is some mapping \( \phi \) such that \( K(x, z) = \phi(x)^T \phi(z) \).
- 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.
- If \( K \) is a valid kernel then

\[
k_{ij} = k(x_i, x_j) = \phi(x_i)^T \phi(x_j) = \phi(x_j)^T \phi(x_i) = k(x_j, x_i) = k_{ji}
\]

- Thus \( K \) is symmetric. It can also be shown that \( K \) is positive semi-definite (show it.).
- 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 $\{x_1, x_2, \ldots, x_N\}$, $(N > 1)$ the corresponding kernel matrix is symmetric positive semi-definite.

- Some valid kernel functions
  - Polynomial kernels
    \[
    K(x, z) = (x^T z + 1)^p
    \]
    $p$ 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 $\sigma$ is specified by the user. This kernel corresponds to an infinite dimensional mapping $\phi$.
  - Sigmoid Kernel
    \[
    K(x, z) = \tanh (\beta_0 x^T z + \beta_1)
    \]
    This kernel only meets Mercer's condition for certain values of $\beta_0$ and $\beta_1$. 

Hamid Beigy  (Sharif University of Technology)
Advantages and disadvantages of SVM

Advantages

- The problem doesn’t have local minima and we can found 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.

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.
<table>
<thead>
<tr>
<th></th>
<th>Table of contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Linear classifiers through origin</td>
</tr>
<tr>
<td>3</td>
<td>Perceptron algorithm</td>
</tr>
<tr>
<td>4</td>
<td>Support vector machines</td>
</tr>
<tr>
<td>5</td>
<td>Lagrangian optimization</td>
</tr>
<tr>
<td>6</td>
<td>Support vector machines (cont.)</td>
</tr>
<tr>
<td>7</td>
<td>Non-linear support vector machine</td>
</tr>
<tr>
<td>8</td>
<td><strong>Generalized linear classifier</strong></td>
</tr>
<tr>
<td>9</td>
<td>Linear discriminant analysis</td>
</tr>
</tbody>
</table>
In nonlinear SVM, we saw that the mapping transformed the problem to a linearly separable one.

Assume that feature vectors to be in the $D$–dimensional space $\mathbb{R}^D$ and they belong to either of the two classes $C_1$ and $C_2$, which are nonlinearly separable.

Let $f_1(\cdot), f_2(\cdot), \ldots, f_k(\cdot)$ be nonlinear functions

$$f_i : \mathbb{R}^D \rightarrow \mathbb{R} \quad \text{for } i = 1, 2, \ldots, k$$

We define this mapping as

$$y = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_k(x) \end{bmatrix}$$

The goal now is to investigate whether there is an appropriate value for $k$ and functions $f_i$ so that classes $C_1, C_2$ are linearly separable in the $k$–dimensional space of the vectors $y$. 
We investigate whether there exists a $k$–dimensional space where we can construct a hyperplane $w \in \mathbb{R}^k$ so that

$$w_0 + w^T y > 0 \quad \text{if} \quad x \in C_1$$
$$w_0 + w^T y < 0 \quad \text{if} \quad x \in C_2$$

If in the original space, two classes were separable by a (nonlinear) hypersurface $g(x) = 0$, then above relations are equivalent to approximating the nonlinear $g(x)$ as a linear combination of $f_i(x)$, that is,

$$g(x) = w_0 + \sum_{i=1}^{k} w_i f_i(x)$$

Once the functions $f_i$ have been selected, the problem becomes a typical design of a linear classifier, that is, to estimate the weights $w_i$ in the $k$–dimensional space.

This justifies the term generalized linear classification.
The generalized linear classification can be shown as.

This diagram corresponds to some previously studied models such as support vector models, linear regression, and neural networks.

Although the model is similar to support vector machines, but the training phase is completely different.
<table>
<thead>
<tr>
<th>1</th>
<th>Introduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Linear classifiers through origin</td>
</tr>
<tr>
<td>3</td>
<td>Perceptron algorithm</td>
</tr>
<tr>
<td>4</td>
<td>Support vector machines</td>
</tr>
<tr>
<td>5</td>
<td>Lagrangian optimization</td>
</tr>
<tr>
<td>6</td>
<td>Support vector machines (cont.)</td>
</tr>
<tr>
<td>7</td>
<td>Non-linear support vector machine</td>
</tr>
<tr>
<td>8</td>
<td>Generalized linear classifier</td>
</tr>
<tr>
<td>9</td>
<td><strong>Linear discriminant analysis</strong></td>
</tr>
</tbody>
</table>
One way to view a linear classification model is in terms of dimensionality reduction.

Consider a two-class problem and suppose we take a $D$–dimensional input vector $x$ and project it down to one dimension using

$$z = W^T x$$

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.

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.

However, by adjusting the components of the weight vector $W$, we can select a projection that maximizes the class separation.

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

$$m_j = \frac{1}{N_j} \sum_{i \in C_j} x_i$$
The simplest measure of the separation of the classes, when projected onto $W$, is the separation of the projected class means.

This suggests that we might choose $W$ so as to maximize

$$m_2 - m_1 = W^T(m_2 - m_1)$$

where

$$m_j = W^T m_j$$

This expression can be made arbitrarily large simply by increasing the magnitude of $W$.

To solve this problem, we could constrain $W$ to have unit length, so that $\sum_i w_i^2 = 1$.

Using a Lagrange multiplier to perform the constrained maximization, we then find that

$$W \propto (m_2 - m_1)$$
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.

This difficulty arises from the strongly non-diagonal covariances of the class distributions.

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.
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.

The projection $z = W^T x$ transforms the set of labeled data points in $x$ into a labeled set in the one-dimensional space $z$.

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

$$z_i = w^T x_i.$$

We can define the total within-class variance for the whole data set to be $s_1^2 + s_2^2$.

The Fisher criterion is defined to be the ratio of the between-class variance to the within-class variance and is given by

$$J(W) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$
• Between-class covariance matrix equals to

\[ S_B = (m_2 - m_1)(m_2 - m_1)^T \]

• Total within-class covariance matrix equals to

\[ S_W = \sum_{i \in C_1} (x_i - m_1)(x_i - m_1)^T + \sum_{i \in C_2} (x_i - m_2)(x_i - m_2)^T \]

• We have

\[
(m_1 - m_2)^2 = \left( W^T m_1 - W^T m_2 \right)^2 \\
= W^T(m_1 - m_2)(m_1 - m_2)^T W \\
= W^T S_B W,
\]
Linear discriminant analysis (cont.)

- Also we have

\[
\begin{align*}
    s_1^2 &= \sum_i \left( W^T x_i - m_2 \right)^2 t_i \\
    &= \sum_i W^T (x_i - m_1)(x_i - m_1)^2 W t_i \\
    &= W^T \left[ \sum_i ((x_i - m_1)(x_i - m_1)^2 \right] W \\
    &= W^T S_1 W,
\end{align*}
\]

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

\[
J(w) = \frac{W^T S_B W}{W^T S_W W}
\]

- Derivative of \( J(W) \) with respect to \( W \) equals to (using \( \frac{\partial x^T A x}{\partial x} = (A + A^T)x \))

\[
\frac{\partial J(W)}{\partial W} = \left[ S_B + S_B^T \right] WW^T S_W W - \left[ S_W + S_W^T \right] WW^T S_B W \\
(W^T S_W W)^2
\]
Linear discriminant analysis (cont.)

- Since $S_B$ and $S_W$ are symmetric, we obtain

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

$$= \frac{2S_B WW^T S_W W - 2S_W WW^T S_B W}{(W^T S_W W)^2}$$

- Both $K_1 = W^T S_W W$ and $K_2 = W^T S_B W$ are scalar

$$\frac{\partial J(W)}{\partial W} = \frac{2S_B WK_1 - 2S_W WK_2}{(K_1)^2}$$

- Since we only find the direction of $W$, the magnitude is not important

$$\frac{\partial J(W)}{\partial W} = 2S_B WK_1 - 2S_W WK_2 = 0$$

- Hence, we obtain

$$S_B W = \frac{K_2}{K_1} S_W W$$

$$= \frac{W^T S_B W}{W^T S_W W} S_W W$$

$$= \lambda S_W W$$
If $S_W$ is invertible, we have

$$S_W^{-1}S_B W = \lambda W$$

Since

$$S_B W = (m_2 - m_1)(m_2 - m_1)^T W = (m_2 - m_1)(m_2 - m_1)$$

Using the above two equations, we obtain

$$\lambda W = S_W^{-1}(m_2 - m_1)(m_2 - m_1)$$

$$W \propto S_W^{-1}(m_2 - m_1)$$

The result $W \propto S_W^{-1}(m_2 - m_1)$ is known as Fisher’s linear discriminant, although strictly it is not a discriminant but rather a specific choice of direction for projection of the data down to one dimension.

The above idea can be extended to multiple classes (Read section 4.1.6 of Bishop).

How the Fisher’s linear discriminant can be used for dimensionality reduction?