Outline

• Support Vector Machines
Support Vector Machines

Key points & concepts

Key points

1. Maximizing the margin
2. Duality optimization task
3. Kernels

Key concepts

1. Hyperplane
2. Dot product
3. Quadratic programming
Support Vector Machines
Key idea for binary classification

We find a hyperplane that separates the two classes in the feature space.

If it is not possible

- allow some training errors, or
- enrich the feature space so that finding a separating hyperplane is possible

A hyperplane is an \((n - 1)\)-dimensional linear subspace of an \(n\) dimensional vector space

\[
\theta_0 + \theta_1 x_1 + \cdots + \theta_m x_m = 0
\]
Hyperplane

The equation $\theta_0 + \Theta^T x = 0$ represents a hyperplane in a two-dimensional space $A_1$ and $A_2$, with a solution set of points where the equation holds true. The hyperplane is illustrated with red and blue dots indicating different sets of solutions or data points.
Recall the classification rule using $\theta_0 + \theta_1 x_1 + \cdots + \theta_m x_m = 0$

$$f(x) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 x_1 + \cdots + \theta_m x_m \geq 0 \\ 0 & \text{if } \theta_0 + \theta_1 x_1 + \cdots + \theta_m x_m < 0 \end{cases}$$

We can use the classification rule for different values of threshold. Here threshold = 0.
Support Vector Machines

Dot product

\begin{itemize}
  \item \( \mathbf{u} = \langle u_1, \ldots, u_m \rangle, \quad \mathbf{v} = \langle v_1, \ldots, v_m \rangle \)
  \item algebraic definition \( \mathbf{u} \cdot \mathbf{v} = u_1 v_1 + \cdots + u_m v_m \)
  \item geometric definition \( \mathbf{u} \cdot \mathbf{v} = ||\mathbf{u}|| \cdot ||\mathbf{v}|| \cdot \cos \alpha \)
\end{itemize}
Quadratic programming optimizes a quadratic objective function subject to constraints. 

I.e. minimize/maximize

\[ f(x), \ x \in \mathcal{D} \]

subject to

\[ g_i(x) \leq 0, \ i = 1, \ldots, G \]
\[ h_j(x) = 0, \ j = 1, \ldots, H \]
Training examples \( D = \{ \langle x_i, y_i \rangle, x_i \in X, y_i \in \{-1, +1\} \} \)

Hyperplane \( g: \theta_0 + \Theta^\top x = 0 \)

- **Margin of** \( x \) w.r.t. \( g \) is distance of \( x \) to \( g \)
  \[
  \rho_g(x) = \frac{|\theta_0 + \Theta^\top x|}{||\Theta||}
  \]

- **Functional margin of** \( \langle x, y \rangle \) w.r.t. \( g \) is
  \[
  \bar{\rho}_g(x, y) = y(\theta_0 + \Theta^\top x)
  \]

- **Geometric margin of** \( \langle x, y \rangle \) w.r.t. \( g \) is functional margin scaled by \( ||\Theta|| \)
  \[
  \rho_g(x, y) = \bar{\rho}_g(x, y)/||\Theta||
  \]
Support Vector Machines
Margins

\[ g : \theta_0 + \Theta^\top x = 0 \]
Support Vector Machines

Margins

- **Functional margin of** $D$ **w.r.t.** $g$

$$\overline{\rho}_g(D) = \min_{\langle x, y \rangle \in D} \overline{\rho}_g(x, y)$$

- **Geometric margin of** $D$ **w.r.t.** $g$

$$\rho_g(D) = \min_{\langle x, y \rangle \in D} \rho_g(x, y)$$
Data set $D = \{\langle x_i, y_i \rangle, x_i \in X, y_i \in \{-1, +1\}\}$ is **linearly separable** if there exists a hyperplane $g : \theta_0 + \Theta^\top x = 0$ that separates the two classes completely, i.e.

$$\forall \langle x, y \rangle \in D : \quad \rho_g(x, y) > 0$$
Support Vector Machines
Binary classification task $Y = \{+1, -1\}$

1. Large margin classifier (linear separability)
2. Soft margin classifier (not linear separability)
3. Kernels (non-linear class boundaries)
Optimization task

$$g^* = \arg\max_g \rho_g(D)$$
Large Margin Classifier
Training data is linearly separable

\[ g : \theta_0 + \Theta^\top x = 0 \]
Training data is linearly separable

\[ \Theta_0 + \Theta^T x \] and \[ k\Theta_0 + (k\Theta)^T x \] define the same hyperplane.

\[
\frac{y_i(\Theta_0 + \Theta^T x_i)}{||\Theta||} = \frac{y_i(k\Theta_0 + (k\Theta)^T x_i)}{||k\Theta||}
\]

Therefore we can scale \( \Theta \) so that \( \rho_g(D) = 1 \).

Then

\[
g^* = \arg\max_g \rho_g(D) = \max_\Theta \frac{1}{||\Theta||}
\]
Large Margin Classifier

Training data is linearly separable

Supporting hyperplanes $\theta_0 + \Theta^\top x = -1$ and $\theta_0 + \Theta^\top x = +1$.

Support vectors are the instances touching the supporting hyperplanes.
Large Margin Classifier
Training data is linearly separable

Optimization task

Minimize

$$\frac{1}{2} ||\Theta||^2$$

subject to

$$y_i(\theta_0 + \Theta^T x_i) \geq 1, \ i = 1, \ldots, n$$

quadratic function and linear constraints $\rightarrow$ quadratic programming
Use the method of Langrangian multipliers to find a minimum of a function subject to constraints.

Define the Lagrangian function $\mathcal{L}_P(\Theta, \Theta_0, \alpha)$

\[
\mathcal{L}_P(\Theta, \theta_0, \alpha) = \frac{1}{2}\|\Theta\|^2 - \sum_{i=1}^{n} \alpha_i(y_i(\theta_0 + \Theta^\top x_i) - 1)
\]

(1)

$\alpha_i \geq 0, i = 1, \ldots, n$
**Primal Lagrangian problem**

\[
\begin{align*}
\text{minimize}_{\Theta, \theta_0} & \text{maximize}_\alpha L_P(\Theta, \theta_0, \alpha) \\
\text{maximize}_\alpha & \text{minimize}_{\Theta, \theta_0} L_P(\Theta, \theta_0, \alpha)
\end{align*}
\]  

(2)

Swap minimize and maximize (see Slater’s condition)

\[
\begin{align*}
\text{maximize}_\alpha & \text{minimize}_{\Theta, \theta_0} L_P(\Theta, \theta_0, \alpha)
\end{align*}
\]  

(3)

subject to

\[\alpha_i \geq 0, i = 1, \ldots, n\]

1. Minimize \(L_P\) w.r.t. \(\Theta\)
   Therefore differentiate \(L_P\) w.r.t. \(\Theta\) and \(\frac{\partial L}{\partial \Theta} = 0\). It gives

\[
\Theta = \sum_{i=1}^{n} \alpha_i y_i x_i
\]  

(4)

2. Minimize \(L_P\) w.r.t. \(\theta_0\)
   Therefore differentiate \(L_P\) w.r.t. \(\theta_0\) and \(\frac{\partial L}{\partial \theta_0} = 0\). It gives \(\sum_{i=1}^{n} \alpha_i y_i = 0\)
3. Substitute (4) into the primal form (1) and solve **Wolfe dual optimization** problem

\[
\max_{\alpha} \mathcal{L}_D(\alpha) = \max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i \cdot x_j
\]

subject to

\[
\alpha_i \geq 0, \ i = 1 \ldots n
\]

\[
\sum_{i=1}^{n} \alpha_i y_i = 0
\]

4. Get \( \alpha^* \)

5. Get \( \Theta^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i \). Assume that \( x_i \) is a support vector. Then

\[
1 = y_i (\theta_0^* + \Theta^* x_i) \rightarrow \theta_0^* = y_i - \Theta^* x_i
\]
Large Margin Classifier

Quadratic programming

• $\alpha^*$ is the solution to the dual problem
• $\Theta^*$ is the solution to the primal problem
• the solutions $\alpha^*$ and $\Theta^*$ must satisfy the Karush-Kuhn-Tucker conditions where one of them is *KKT dual complementarity*:

$$\alpha^*_i \cdot (1 - y_i (\theta^*_0 + \Theta^*_\top x_i)) = 0$$

• $y_i (\theta^*_0 + \Theta^*_\top x_i) \neq 1$, i.e., $x_i$ is not support vector $\Rightarrow \alpha^*_i = 0$
• $\alpha^*_i \neq 0 \Rightarrow y_i (\theta^*_0 + \Theta^*_\top x_i) = 1$, i.e., $x_i$ is support vector

I.e., finding $\Theta^*$ is equivalent to finding support vectors and their weights
**Prediction** for a new instance $\mathbf{x}$

$$f(\mathbf{x}) = \text{sign} \left( \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i \cdot \mathbf{x} + \theta_0^* \right)$$

- similarity between $\mathbf{x}$ and support vector $\mathbf{x}_i$: a support vector that is more similar contributes more to the classification
- support vector that is more important, i.e. has larger $\alpha_i$, contributes more to the classification
- if $y_i$ is positive, than the contribution is positive, otherwise negative
In a real problem it is unlikely that a hyperplane will exactly separate the data – even if a curved decision boundary is possible. So exactly separating the data is probably not desirable – if the data has noise and outliers, a smooth decision boundary that ignores a few data points is better than one that loops around the outliers.

Therefore

\[
\text{minimize } ||\Theta||^2 \text{ AND the number of training mistakes}
\]
Soft Margin Classifier

Training data is not linearly separable
Soft Margin Classifier
Slack variables

\[\xi_i \geq 0\]

- \(\xi_i = 0\) if \(x_i\) is correctly classified

- \(\xi_i\) is distance to \(y_i\)'s supporting hyperplane otherwise
  - margin violation – \(0 < \xi_i \leq 1/\|\Theta\|\), see \(\xi_1, \xi_3\) above
  - misclassification – \(\xi_i > 1/\|\Theta\|\), see \(\xi_2\) above
Soft Margin Classifier
Optimization problem

Minimize
\[ \frac{1}{2} \| \Theta \|^2 + C \sum_{i=1}^{n} \xi_i \]

subject to
\[ \xi_i \geq 0, y_i(\theta_0 + \Theta^T x_i) \geq 1 - \xi_i, \quad i = 1, \ldots, n \]

\( C \geq 0 \) trade-off parameter

- small \( C \) \( \Rightarrow \) large margin
  relaxed model; misclassifications are not penalized

- large \( C \) \( \Rightarrow \) narrow margin
  misclassifications are penalized strongly
  the model will not generalize much
For each constraint $y_i(\theta_0 + \Theta^T x_i) \geq 1 - \xi_i$
introduce Lagrange multiplier $\alpha_i \geq 0$.

Let $\alpha = \langle \alpha_1, ..., \alpha_n \rangle$.

Primal Lagrangian $L_P(\Theta, \theta_0, \xi, \alpha)$ is given by

$$L_P(\Theta, \theta_0, \xi, \alpha) = \frac{1}{2}||\Theta||^2 + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i(y_i(\theta_0 + \Theta^T x_i) - 1 + \xi_i)$$  \hspace{1cm} (5)
Primal Lagrangian problem

$$\text{minimize}_{\Theta, \theta_0, \xi} \text{maximize}_\alpha \mathcal{L}_P(\Theta, \theta_0, \xi, \alpha)$$ (6)

$$\text{maximize}_\alpha \text{minimize}_{\Theta, \theta_0, \xi} \mathcal{L}_P(\Theta, \theta_0, \xi, \alpha)$$ (7)

subject to

$$\alpha_i \geq 0, \xi_i \geq 0, i = 1, \ldots, n$$

1. Minimize $\mathcal{L}_P$ w.r.t. $\Theta$. Therefore $\frac{\partial \mathcal{L}}{\partial \Theta} = 0$. It gives

$$\Theta = \sum_{i=1}^{n} \alpha_i y_i x_i$$ (8)

2. Minimize $\mathcal{L}_P$ w.r.t. $\theta_0$. Therefore $\frac{\partial L}{\partial \theta_0} = 0$. It gives

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$ (9)
3. Minimize $\mathcal{L}_P$ w.r.t. $\xi$. Therefore $\frac{\partial \mathcal{L}}{\partial \xi} = 0$. It gives

$$C\xi - \alpha = 0$$  

(10)

4. Substitute (8) into the primal form (5) and solve Wolfe dual opt. problem

$$\max_{\alpha} \mathcal{L}_D(\alpha) = \max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^{n} \alpha_i \alpha_j y_i y_j x_i x_j$$

subject to

$$\alpha_i \geq 0, \quad i = 1, \ldots, n$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

5. Get $\alpha^*$

6. Get $\Theta^*$
**Prediction** for a new instance $\mathbf{x}$

$$f(\mathbf{x}) = \text{sign}(\sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i \cdot \mathbf{x} + \theta_0^*)$$
Non-linear boundary

If the examples are separated by a nonlinear region
Recall polynomial regression

Polynomial regression is an extension of linear regression where the relationship between features and target value is modelled as a $d$-th order polynomial.

**Simple regression**

\[ y = \Theta_0 + \Theta_1 x_1 \]

**Polynomial regression**

\[ y = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_1^2 + \ldots \Theta_d x_1^d \]

It is still a linear model with features $A_1, A_1^2, \ldots, A_1^d$.

This defines a feature mapping $\phi(x_1) = [x_1, x_1^2, \ldots, x_1^d]$
Non-linear boundary

• for example $\phi(x_1) = [x_1 - 5, (x_1 - 5)^2]$
**Idea**

- Apply Large/Soft margin classifier not to the original features but to the features obtained by the feature mapping \( \phi(x) : \mathcal{R}^m \rightarrow \mathcal{F} \).

- Large/Soft margin classifier uses dot product \( x_i \cdot x_j \). Replace it with \( \phi(x_i) \cdot \phi(x_j) \).
$\phi : (x_1, x_2) \rightarrow (\sqrt{2}x_1x_2, x_2^2)$

$$\left(\frac{x_1}{a}\right)^2 + \left(\frac{x_2}{b}\right)^2 = 1 \rightarrow \frac{x_1^2}{a^2} + \frac{x_2^3}{b^2} = 1$$

Source: http://omega.albany.edu:8008/machine-learning-dir/notes-dir/ker1/ker1-l.html
Kernels

However, finding $\phi$ could be expensive.

**Kernel trick**

- No need to know what $\phi$ is and what the feature space is, i.e. no need to explicitly map the data to the feature space
- Define a kernel function $K : \mathcal{R}^m \times \mathcal{R}^m \rightarrow \mathcal{R}$
- Replace the dot product $\mathbf{x}_i \cdot \mathbf{x}_j$ with a Kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$:

$$
\mathcal{L}_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)
$$
Prediction for a new instance $x$

$$f(x) = \text{sign} \left( \sum_{i=1}^{n} \alpha_i^* y_i K(x_i, x) + \theta_0^* \right)$$
Common Kernel functions

- **Linear**
  \[ K(x_i, x_j) = x_i \cdot x_j \]

- **Polynomial**
  \[ K(x_i, x_j) = (\gamma x_i \cdot x_j + c)^d \]
  - smaller degree can generalize better
  - higher degree can fit (only) training data better

- **Radial basis function**
  \[ K(x_i, x_j) = \exp(-\gamma(\|x_i - x_j\|)^2)) \]
  - very robust
  - use it when polynomial kernel is weak to fit data

- **Sigmoid**
  \[ K(x_i, x_j) = \tanh(\gamma x_i \cdot x_j + c), \text{ where } \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \]
Radial Basis Function Kernel

- $K(x_i, l) = e^{-\gamma \|x_i - l\|^2}$

Source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html
Regularized logistic regression

\[ f(x) = \frac{1}{1 + e^{-\Theta^T x}} \]

\[ L(\Theta) = -\sum_{i=1}^{n} y_i \log P(y_i|x_i; \Theta) + (1 - y_i) \log(1 - P(y_i|x_i; \Theta)) \]

\[ \Theta^*_R = \arg\min_{\Theta} [L(\Theta) + \lambda \cdot \text{penalty}(\Theta)] \]
Regularized logistic regression

Ridge regression

$$\Theta^*_R = \text{arg min}_\Theta - \left[ \sum_{i=1}^{n} y_i \log(f(x_i)) + (1 - y_i) \log(1 - f(x_i)) \right] + \lambda \sum_{j=1}^{m} \theta_j^2 ] = $$

$$= \text{arg min}_\Theta \left[ \sum_{i=1}^{n} y_i (- \log(f(x_i))) + (1 - y_i)(- \log(1 - f(x_i))) \right] + \lambda \sum_{j=1}^{m} \theta_j^2 ] = $$

$$= \text{arg min}_\Theta \left[ \sum_{i=1}^{n} y_i L_1(\Theta) + (1 - y_i) L_0(\Theta) + \lambda \sum_{j=1}^{m} \theta_j^2 \right]$$
Since

\[ A + \lambda B \equiv CA + B, \quad C = \frac{1}{\lambda} \]

then

\[ \Theta_R^* = \arg\min_{\Theta} \left[ \sum_{j=1}^{m} \theta_j^2 + C \left[ \sum_{i=1}^{n} y_i L_1(\Theta) + (1 - y_i) L_0(\Theta) \right] \right] \]

where

\[ L_1(\Theta) = -\log \frac{1}{1 + e^{-\Theta^\top x}} \]

\[ L_0(\Theta) = -\log(1 - \frac{1}{1 + e^{-\Theta^\top x}}) \]
Regularized logistic regression
Ridge regression and Soft Margin Classifier

\[ \Theta^*_R = \arg\min_{\Theta} \left[ \sum_{j=1}^{m} \theta_j^2 + C \sum_{i=1}^{n} \log(1 + e^{-\bar{y}_i \Theta^\top x_i}) \right] \]

where

\[ \bar{y}_i = \begin{cases} 
-1 & \text{if } y_i = 0 \\
+1 & \text{if } y_i = 1 
\end{cases} \]

Soft Margin Classifier is equivalent to the regularization problem:

\[ \Theta^* = \arg\min_{\Theta} \sum_{j=1}^{m} \theta_j^2 + C \sum_{i=1}^{n} \xi_i \]
\[ \Theta^* = \arg\min_{\Theta} \sum_{j=1}^{m} \theta_j^2 + C \sum_{i=1}^{n} \xi_i \]

\( \xi_i \geq 0 \) is equivalent to \( \xi_i = \max(0, 1 - y_i \Theta^T x_i) \), i.e.

\[ \Theta^* = \arg\min_{\Theta} \left[ \sum_{j=1}^{m} \theta_j^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i \Theta^T x_i) \right] \]

s.t. \( \Theta^T x_i \geq 1 - \xi_i \) if \( y_i = +1 \) and \( \Theta^T x_i \leq -1 + \xi_i \) if \( y_i = -1 \)

**Hinge loss** = \( \max(0, 1 - y_i \Theta^T x) \)

1. \( y_i \Theta^T x_i > 1 \): no contribution to loss
2. \( y_i \Theta^T x_i = 1 \): no contribution to loss
3. \( y_i \Theta^T x_i < 1 \): contribution to loss
Soft Margin Classifier

\[ y_i (\theta_0 + \Theta^T x_i) \]
Summary of Examination Requirements

- Hyperplane, margin, functional margin, geometric margin of example and data set
- Large margin classifier
  linearly separable data, supporting hyperplanes, support vectors, optimization task, prediction function
- Soft margin classifier
  not linearly separable data, supporting hyperplanes, support vectors, slack variables, optimization task, hyperparameter $C$, prediction function
- Kernel trick
  feature mapping, Kernel functions, prediction function