Outline

This lecture: Support Vector Machine

- From perceptron to SVM
- Margin
- Max margin classifier
- Lagrange duality
- Kernels

Reference:

- Mustafa, *Learning from Data*, e-Chapter
- Duda-Hart-Stork, *Pattern Classification*, Chapter 5.5
- Chris Bishop, *Pattern Recognition*, Chapter 7.1
- UCSD Statistical Learning
  http://www.svcl.ucsd.edu/courses/ece271B-F09/
- Stanford CS 229
Support Vector Machine

SVM is the solution of this optimization

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| \mathbf{w} \|^2_2 \\
\text{subject to} & \quad y_j (\mathbf{w}^T \mathbf{x}_j + w_0) \geq 1, \quad j = 1, \ldots, N.
\end{align*}
\]
Dual SVM

- **Primal**

  \[
  \text{minimize}_{\mathbf{w}, w_0} \quad \frac{1}{2} \| \mathbf{w} \|_2^2
  \]

  subject to \( y_j (\mathbf{w}^T \mathbf{x}_j + w_0) \geq 1, \quad j = 1, \ldots, N. \)

- **Strong Duality**

  \[
  \min_{\mathbf{w}, w_0} \max_{\lambda \geq 0} L(\mathbf{w}, w_0, \lambda) = \max_{\lambda \geq 0} \min_{\mathbf{w}, w_0} L(\mathbf{w}, w_0, \lambda)
  \]

  \[
  \text{primal} \quad \text{dual}
  \]

- **Dual**

  \[
  \text{maximize}_{\lambda \geq 0} \quad -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{j=1}^{N} \lambda_j
  \]

  subject to \( \sum_{j=1}^{N} \lambda_j y_j = 0. \)
Interpreting SVM Solution

- The weights are computed as

\[ \mathbf{w}^* = \sum_{j=1}^{N} \lambda_j^* y_j \mathbf{x}_j. \]

- This is support vector: \( \lambda_j \) is either \( \lambda_j = 0 \) or \( \lambda_j > 0 \).
- Pick any support vector \( \mathbf{x}^+ \in C_+ \) and \( \mathbf{x}^- \in C_- \).
- Then we have

\[ \mathbf{w}^T \mathbf{x}^+ + w_0 = +1, \quad \text{and} \quad \mathbf{w}^T \mathbf{x}^- + w_0 = -1. \]

- Sum them, we have \( \mathbf{w}^T (\mathbf{x}^+ + \mathbf{x}^-) + 2w_0 = 0 \), which means

\[ w_0^* = - \frac{(\mathbf{x}^+ + \mathbf{x}^-)^T \mathbf{w}^*}{2} \]
Interpreting SVM Solution
We can consider this problem

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| \mathbf{w} \|^2_2 \\
\text{subject to} & \quad y_j (\mathbf{w}^T \mathbf{x}_j + w_0) \geq 1 - \xi_j, \\
& \quad \xi_j \geq 0, \quad \text{for} \quad j = 1, \ldots, n,
\end{align*}
\]

But we need to control \( \xi \), for otherwise the solution will be \( \xi = \infty \).

How about this:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| \mathbf{w} \|^2_2 + C \| \xi \|^2 \\
\text{subject to} & \quad y_j (\mathbf{w}^T \mathbf{x}_j + w_0) \geq 1 - \xi_j, \\
& \quad \xi_j \geq 0, \quad \text{for} \quad j = 1, \ldots, n,
\end{align*}
\]

Control the energy of \( \xi \).
Role of $C$

- If $C$ is big, then we enforce $\xi$ to be small.
- If $C$ is small, then $\xi$ can be big.
No Misclassification?

- You can have misclassification in soft SVM
- \( \xi_j \) can be big for a few outliers

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| w \|_2^2 + C \| \xi \|_2^2 \\
\text{subject to} & \quad y_j (w^T x_j + w_0) \geq 1 - \xi_j, \\
& \quad \xi_j \geq 0, \quad \text{for } j = 1, \ldots, N.
\end{align*}
\]
Non-Separable

Not so bad. Soft margin can handle.
Non-Separable

Seriously?!
A Closer Look at Nonlinearity

- Each vector is \( \mathbf{x} = [1, \ x_1, \ x_2]^T \).
- The discriminant function is
  \[
g(\mathbf{x}) = 0.5 - (x_1^2 + x_2^2).
  \]
- The hypothesis function is
  \[
h(\mathbf{x}) = \text{sign} \left( 0.5 - x_1^2 - x_2^2 \right).
  \]
Input Space and Feature Space

Input space $\mathcal{X}$
- $x = [1, x_1, x_2]^T$: input vector.
- Classified using a nonlinear hypothesis
  \[ h(x) = \text{sign} \left( 0.5 - x_1^2 - x_2^2 \right) \]

Feature space $\mathcal{Z}$
- $z = [1, x_1^2, x_2^2]^T$: feature vector.
- Classified using a linear hypothesis
  \[ h(z) = \text{sign} \left( 0.5 - z_1 - z_2 \right) \]
- The dimensionality of $z$ does not need to be the same as $x$.
- Could be smaller.
- Could be bigger; Could be infinite dimension.
Define \( \mathbf{x} = [1, x_1, x_2]^T \).

Define \( \mathbf{z} = [1, x_1^2, x_2^2]^T \).

Then, \( h(\mathbf{x}) \) is

\[
\begin{align*}
   h(\mathbf{x}) &= \text{sign} \left( 0.5 - x_1^2 - x_2^2 \right) \\
   &= \text{sign} \left( \begin{bmatrix} 0.5 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ x_1^2 \\ x_2^2 \end{bmatrix} \right) \\
   &= \text{sign} \left( \begin{bmatrix} \tilde{w}_0 & \tilde{w}_1 & \tilde{w}_2 \end{bmatrix} \begin{bmatrix} 1 \\ z_1 \\ z_2 \end{bmatrix} \right) \\
   &= \text{sign} \left( \mathbf{w}^T \mathbf{z} + \tilde{w}_0 \right)
\end{align*}
\]

Nonlinear in \( \mathbf{x} \), but linear in \( \mathbf{z} \)!
Non-Linear Transform

Classifier is

\[ h(x) = \text{sign}(w^T \Phi(x) + w_0) \]

\[ z = \Phi(x) = \begin{bmatrix} 1 \\ x_1^2 \\ x_2^2 \end{bmatrix} \]
Non-linear Transform for SVM

Here is one possibility:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| \mathbf{w} \|_2^2 \\
\text{subject to} & \quad y_j (\mathbf{w}^T \Phi(\mathbf{x}_j) + w_0) \geq 1, \quad j = 1, \ldots, N.
\end{align*}
\]

But you need to know \( \Phi \)

\( \Phi \) could be very hard to get

Actually, you do not need \( \Phi \)
The Kernel Trick

- A trick to turn linear classifier to nonlinear classifier.
- Dual SVM

\[
\text{maximize } \lambda \geq 0 \quad \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j x_i^T x_j + \sum_{j=1}^{n} \lambda_j \\
\text{subject to } \sum_{j=1}^{n} \lambda_j y_j = 0.
\]

- Kernel Trick

\[
\text{maximize } \lambda \geq 0 \quad \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \Phi(x_i)^T \Phi(x_j) + \sum_{j=1}^{n} \lambda_j \\
\text{subject to } \sum_{j=1}^{n} \lambda_j y_j = 0.
\]

- You have to do this in dual. Primal is hard. See next slide.
The Kernel Trick

- Define

\[ K(x_i, vx_j) = \Phi(x_i)^T \Phi(x_j). \]

- The matrix \( Q \) is

\[
Q = \begin{bmatrix}
    y_1 y_1 x_1^T x_1 & \ldots & y_1 y_N x_1^T x_N \\
    y_2 y_1 x_2^T x_1 & \ldots & y_2 y_N x_2^T x_N \\
    \vdots & \vdots & \vdots \\
    y_N y_1 x_N^T x_1 & \ldots & y_N y_N x_N^T x_N
\end{bmatrix}
\]

- By Kernel Trick:

\[
Q = \begin{bmatrix}
    y_1 y_1 K(x_1, x_1) & \ldots & y_1 y_N K(x_1, x_N) \\
    y_2 y_1 K(x_2, x_1) & \ldots & y_2 y_N K(x_2, x_N) \\
    \vdots & \vdots & \vdots \\
    y_N y_1 K(x_N, x_1) & \ldots & y_N y_N K(x_N, x_N)
\end{bmatrix}
\]
Kernel

- The inner product $\Phi(x_i)^T \Phi(x_j)$ is called a kernel
  \[ K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j). \]

- Second-Order Polynomial kernel
  \[ K(u, v) = (u^T v)^2. \]

- Degree-Q Polynomial kernel
  \[ K(u, v) = (\gamma u^T v + c)^Q. \]

- Gaussian Radial Basis Function (RBF) Kernel
  \[ K(u, v) = \exp \left\{ -\frac{\|u - v\|^2}{2\sigma^2} \right\}. \]
Can We Find $\Phi$ from $K$?

Example. Second Order Polynomial

$$K(u, v) = (u^T v)^2, \quad u, v \in \mathbb{R}^2.$$ 

If we write out $K(u, v)$, we can show that

$$K(u, v) = \left( \sum_{i=1}^{2} u_i v_i \right)^2 = \left( \sum_{i=1}^{2} u_i v_i \right) \left( \sum_{j=1}^{2} u_j v_j \right)$$

$$= \sum_{i=1}^{2} \sum_{j=1}^{2} (u_i u_j)(v_i v_j)$$

$$= \begin{bmatrix} u_1 u_1 & u_1 u_2 & u_2 u_1 & u_2 u_2 \end{bmatrix} \begin{bmatrix} v_1 & v_1 & v_2 & v_1 \\ v_1 & v_2 & v_1 & v_2 \\ v_2 & v_1 & v_2 & v_2 \end{bmatrix} = \Phi(u)^T \Phi(v).$$
Can We Find $\Phi$ from $K$?

**Example.** Second Order Polynomial

$$K(u, v) = (u^Tv)^2, \quad u, v \in \mathbb{R}^2.$$ 

We just showed that

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = z = \Phi(v) = \Phi \left( \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \right) = \begin{bmatrix} v_1 v_1 \\ v_1 v_2 \\ v_2 v_1 \\ v_2 v_2 \end{bmatrix}$$
SVM with Second Order Kernel

Boxed samples = Support vectors.
What if $d$ grows?

**Example.** Second Order Polynomial

$$K(u, v) = (u^T v)^2, \quad u, v \in \mathbb{R}^d.$$ 

We just showed that

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{d^2} \end{bmatrix} = z = \Phi(v) = \Phi \begin{bmatrix} v_1 \\ \vdots \\ v_d \end{bmatrix} = \begin{bmatrix} v_1 v_1 \\ v_1 v_2 \\ \vdots \\ v_d v_d \end{bmatrix}$$

- Dimension of $z$ grows quadratically as $d$ increase!
- Nightmare ... For every $x_j$ you need to compute $\Phi(x_j)$.
- Good news: In dual SVM, you just need to compute $K(x_i, x_j)$.
- You do not need $\Phi(x_i)$ and $\Phi(x_j)$. 
Example. Radial Basis Function

\[ K(u, v) = \exp \left\{ -\gamma \| u - v \|^2 \right\} . \]

- Typical \( \gamma \in [0, 1] \).
- \( \gamma \) too big: Over-fit.
Non-Linear Transform for RBF?

- Let us consider scalar \( u \in \mathbb{R} \).

\[
K(u, v) = \exp\{-(u - v)^2\} = \exp\{-u^2\} \exp\{2uv\} \exp\{-v^2\}
\]

\[
= \exp\{-u^2\} \left( \sum_{k=0}^{\infty} \frac{2^k u^k v^k}{k!} \right) \exp\{-v^2\}
\]

\[
= \exp\{-u^2\} \left( 1, \sqrt{\frac{2^1}{1!}} u, \sqrt{\frac{2^2}{2!}} u^2, \sqrt{\frac{2^3}{3!}} u^3, \ldots, \right)^T
\]

\[
\times \left( 1, \sqrt{\frac{2^1}{1!}} v, \sqrt{\frac{2^2}{2!}} v^2, \sqrt{\frac{2^3}{3!}} v^3, \ldots, \right) \exp\{-v^2\}
\]

- So \( \Phi \) is

\[
\Phi(x) = \exp\{-x^2\} \left( 1, \sqrt{\frac{2^1}{1!}} x, \sqrt{\frac{2^2}{2!}} x^2, \sqrt{\frac{2^3}{3!}} x^3, \ldots, \right)
\]
So You Need

**Example.** Radial Basis Function

\[ K(u, v) = \exp \left\{ -\gamma \| u - v \|^2 \right\}. \]

The non-linear transform is:

\[ \Phi(x) = \exp\{-x^2\} \left(1, \sqrt{\frac{2^1}{1!}}x, \sqrt{\frac{2^2}{2!}}x^2, \sqrt{\frac{2^3}{3!}}x^3, \ldots, \right) \]

- You need infinite dimensional non-linear transform!
- But to compute the kernel \( K(u, v) \) you do not need \( \Phi \).
- Another Good thing about Dual SVM: You can do infinite dimensional non-linear transform.
- Cost of computing \( K(u, v) \) is bottleneck by \( \| u - v \|^2 \).
Is RBF Always Better than Linear?

- Noisy dataset: Linear works well.
- RBF: Over fit.
How Do We Know $\Phi(x)$ Exists …

for a given $K(u, v)$?

- By Construction
- By Mathematical Properties: Mercer Condition
- Who Cares?
The Mathematical Property

Given \( \{x_j\}_{j=1}^N \), construct a \( N \times N \) matrix \( K \) such that

\[
[K]_{ij} = K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j).
\]

**Claim**: \( K \) is positive semi-definite.

Let \( z \) be an arbitrary vector. Then,

\[
z^T K z = \sum_{i=1}^n \sum_{j=1}^N z_i K_{ij} z_j
= \sum_{i=1}^N \sum_{j=1}^N z_i \Phi(x_i)^T \Phi(x_j) z_j
= \sum_{i=1}^N \sum_{j=1}^N z_i \left( \sum_{k=1}^N [\Phi(x_i)]_k [\Phi(x_j)]_k \right) z_j
\geq 0
\]

where \([\Phi(x_i)]_k\) denotes the \( k \)-th element of the vector \( \Phi(x_i) \).
The Mathematical Property

- We just showed that: If $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$ for any $x_1, \ldots, x_N$, then $K$ is symmetric positive semi-definite.
- The converse also holds: If $K$ is symmetric positive semi-definite for any $x_1, \ldots, x_N$, then there exist $\Phi$ such that $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$.
- The converse if (of course) very hard to prove. Called Mercer Condition.
- Kernels satisfying Mercer’s condition have $\Phi$.
- You can use the condition to rule out invalid kernels. But proving a valid kernel is still hard.
- In practice, who cares.
Testing with Kernels

- Recall:
  \[ w^* = \sum_{n=1}^{N} \lambda_n^* y_n x_n. \]

- The hypothesis function is
  \[
  h(x) = \text{sign} \left( w^*^T x + w_0^* \right)
  = \text{sign} \left( \left( \sum_{n=1}^{N} \lambda_n^* y_n x_n \right)^T x + w_0^* \right)
  = \text{sign} \left( \sum_{n=1}^{N} \lambda_n^* y_n x_n^T x + w_0^* \right).
  \]

- Now you can replace \( x_n^T x \) by \( K(x_n, x) \).
Any Question?