Machine Learning
Lecture: Support Vector Machines

Hiroshi Shimodaira

16 February 2024

Ver. 1.0
Questions you should be able to answer after this week

• What is Support Vector Machine (SVM)?
• Training (optimisation problem) of linear SVM?
  What is maximum margin
• How to solve the optimisation problem?
• What are the support vectors?
• What is soft-margin SVM (SVM with slack variables)?
• How to make non-linear SVM?
• What is kernel and what is kernel trick?
• What are pros and cons with SVM?
• What applications are SVM successful for?
History of machine learning

18c Naive Bayes classifier

1940s Threshold logic - Warren McCulloch and Walter Pitts
Logistic regression - Joseph Berkson

1951 k-NN - Evelyn Fix and Joseph Hodges

1957 Perceptron - Frank Rosenblatt

1959 Decision tree - William Belson (?)

1986 ANN with EBP - D.Rumelhart, G.Hinton, and R.Williams

1993-97 Support Vector Machine - Vladimir Vapnik
Recap – Logistic Regression

• \( P(Y = 1|x) = \frac{1}{1 + \exp(-(w^T x + w_0))} \)

\[ x = (x_1, \ldots, x_d)^T, \]
\[ w = (w_1, \ldots, w_d)^T, Y \in \{0, 1\} \]

• Training on a data set \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \) based on maximum likelihood estimation (MLE):

\[
\max_{w, w_0} \prod_{i=1}^{n} P(Y = y_i|x_i)
\]
Decision boundary and decision regions

\[ P(Y = 1 | x) = \frac{1}{1 + \exp(- (w^T x + w_0))} \rightarrow \text{decision boundary: } w^T x + w_0 = 0 \]
Decision boundary and decision regions (cont.)

\[ P(Y = 1|\mathbf{x}) = \frac{1}{1 + \exp \left( - (\mathbf{w}^T \mathbf{x} + w_0) \right)} \]

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Decision boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>line ( w_1 x_1 + w_2 x_2 + w_0 = 0 )</td>
</tr>
<tr>
<td>3</td>
<td>plane ( w_1 x_1 + w_2 x_2 + w_3 x_3 + w_0 = 0 )</td>
</tr>
<tr>
<td>\vdots</td>
<td></td>
</tr>
<tr>
<td>( d )</td>
<td>hyperplane ( \left( \sum_{i=1}^{d} w_i x_i \right) + w_0 = 0 )</td>
</tr>
</tbody>
</table>
Linear classifiers and large margin classifiers

\[ \hat{y}(x) = f(x) = w^T x + w_0 \]

\[ w^T x_i + w_0 = 0 \]

\[ w^T x_i + w_0 = +1 \]

\[ w^T x_i + w_0 = -1 \]
Large margin classifiers (cont.)

Proposed by several people, e.g. Vladimir Vapnik (1963, 1992)

\[ \mathbf{w}^T \mathbf{x}_i + w_0 \geq +1 \quad \forall i \text{ s.t. } y_i = +1 \]
\[ \mathbf{w}^T \mathbf{x}_i + w_0 \leq -1 \quad \forall i \text{ s.t. } y_i = -1 \]
Margin

\[
\| \mathbf{p}_1 - \mathbf{p}_2 \| = \| \mathbf{p}_1 \| - \| \mathbf{p}_2 \| \\
= \left| \frac{-w_0 + 1}{\| \mathbf{w} \|} - \frac{-w_0 - 1}{\| \mathbf{w} \|} \right| \\
= \frac{2}{\| \mathbf{w} \|} = 2r
\]

where

\[
1 = \mathbf{w}^T \mathbf{p}_1 + w_0 \\
= \| \mathbf{w} \| \| \mathbf{p}_1 \| \cos(\theta)|_{\theta=0} + w_0 \\
= \| \mathbf{w} \| \| \mathbf{p}_1 \| + w_0
\]

\[
\| \mathbf{p}_1 \| = \frac{-w_0 + 1}{\| \mathbf{w} \|}
\]
Support Vector Machine (SVM)

Training
\[
\max_w \frac{1}{\|w\|}
\]

s.t. \(w^T x_i + w_0 \geq +1\) for all \(i\) with \(y_i = +1\)
\(w^T x_i + w_0 \leq -1\) for all \(i\) with \(y_i = -1\)

Equivalent to
\[
\min_w \frac{1}{2} \|w\|^2
\]

s.t. \(y_i (w^T x_i + w_0) \geq 1\) for all \(i\)

NB: constrained, quadratic and convex optimisation problem → no local minima!

Solution: \(w = \sum_{i=1}^{n} \alpha_i y_i x_i\), \(\alpha_i \geq 0\) … most of \(\alpha_i\) are zeros normally

Those \(\{x_i\}\) whose \(\alpha_i > 0\) are called support vectors.

Classification
\[
g(x) = \text{sgn}(w^T x + w_0) = \text{sgn} \left( \sum_{i=1}^{n} \alpha_i y_i x_i^T x + w_0 \right)
\]
Why $+1$ instead of $+\varepsilon$?

Assuming $\varepsilon > 0$,

$$\min_{w, w_0} \frac{1}{2} \|w\|^2$$

subject to

$$y_i (w^T x_i + w_0) \geq \varepsilon \text{ for all } i$$

$$\Rightarrow$$

$$\min_{w, w_0} \frac{1}{2} \|w\|^2$$

subject to

$$y_i \left( \frac{w^T}{\varepsilon} x_i + \frac{w_0}{\varepsilon} \right) \geq 1 \text{ for all } i$$

Letting $\dot{w} = \frac{w}{\varepsilon}$ and $\dot{w}_0 = \frac{w_0}{\varepsilon}$,

$$\min_{w, w_0} \frac{\varepsilon^2}{2} \|\dot{w}\|^2$$

subject to

$$y_i (\dot{w}^T x_i + \dot{w}_0) \geq 1 \text{ for all } i$$
Optimisation problems in SVM

\[
\begin{align*}
\min_{w, w_0} & \quad \frac{1}{2} w^T w \\
\text{s.t.} & \quad y_i (w^T x_i + w_0) \geq 1 \quad \text{for all } i
\end{align*}
\]

Using the Lagrange multipliers \( \alpha_i \geq 0 \), the Lagrangian is given as:

\[
L(\alpha, \dot{w}) = \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i \left( y_i (w^T x_i + w_0) - 1 \right)
\]

where \( \alpha = (\alpha_1, \ldots, \alpha_n) \) and \( \dot{w} = (w, w_0) \). The dual problem is defined as

\[
\max_{\alpha} \quad L(\alpha, \dot{w}) \\
\text{s.t.} & \quad \alpha \geq 0
\]
Optimisation problems in SVM (cont.)

\[ L(\alpha, \dot{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^{n} \alpha_i \left( y_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1 \right) \]

\[ \frac{\partial L(\alpha, \dot{w})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0, \]

\[ \frac{\partial L(\alpha, \dot{w})}{\partial w_0} = -\sum_{i=1}^{n} \alpha_i y_i = 0. \]

\[ \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \]

\[ 0 = \sum_{i=1}^{n} \alpha_i y_i \]
Optimisation problems in SVM (cont.)

Putting the results to the Lagrangian yields:

\[ L(\alpha, \dot{w}) = \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i \left( y_i (w^T x_i + w_0) - 1 \right) \]

\[ = \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^T x_j - \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^T x_j + \sum_{i=1}^{n} \alpha_i \]

\[ = -\frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^T x_j + \sum_{i=1}^{n} \alpha_i \]

The necessary and sufficient conditions for \( w^* \) to be an optimum are:

\[ \frac{\partial L(\alpha^*, \dot{w}^*)}{\partial w} = 0, \quad \frac{\partial L(\alpha^*, \dot{w}^*)}{\partial w_0} = 0, \quad \alpha_i^* \geq 0, \quad y_i (w^T x_i + w_0) - 1 \geq 0, \]

\[ \alpha_i^* \left( y_i (w^T x_i + w_0) - 1 \right) = 0, \text{ for all } i \quad \cdots \text{ Karush-Kuhn-Tuckert (KKT) condition} \]

which means that either \( \alpha_i^* = 0 \) or \( y_i (w^T x_i + w_0) - 1 = 0 \).
**SVM with slack variables – soft margin SVM**

**Hard margin SVM**

\[
\begin{align*}
\text{min} & \quad w^T w \\
\text{s.t.} & \quad y_i(w^T x_i + w_0) \geq 1 \text{ for all } i
\end{align*}
\]

**Soft margin SVM**

\[
\begin{align*}
\text{min} & \quad w^T w + C \left( \sum_{i=1}^{n} \xi_i \right), \quad \text{where } C > 0, \\
\text{s.t.} & \quad y_i(w^T x_i + w_0) \geq 1 - \xi_i \text{ for all } i, \quad \xi_i \geq 0
\end{align*}
\]
Loss function in soft-margin SVM

\[
\min_{w,w_0} \quad w^T w + C \left( \sum_{i=1}^{n} \xi_i \right), \quad \text{where } C > 0,
\]

s.t. \quad y_i (w^T x_i + w_0) \geq 1 - \xi_i \quad \text{for all } i, \quad \xi_i \geq 0

The hinge loss:

\[
\ell(t) = \max(0, 1 - t)
\]

\[
= \begin{cases} 
0, & \text{if } t \geq 1, \\
1 - t, & \text{otherwise,}
\end{cases}
\]

where \( t = y (w^T x + w_0) \).
Non-linear SVM

Training data in input space

SVM decision hyperplane in feature space

SVM decision surface in input space

Linear discriminant function decision surface in input space

\( \Phi \)

\( \Phi^{-1} \)

SVM decision hyperplane in feature space
Non-linear SVM (cont.)

• Conceptual steps to construct a non-linear SVM
  
  Step 1 Transform $x$ to $\phi(x)$ in a high-dimensional space (feature space)
  
  Step 2 Train a SVM in the feature space
  
  Step 3 Classify data in the feature space

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i)^T \phi(x) + w_0$$

• Instead of applying the non-linear transformation and carrying out calculation in the feature space, use a kernel function $k(x_i, x_j)$ such that

$$k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

(cf. 'kernel trick')

$$L(\alpha, \xi) = -\frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j k(x_i, x_j) + \sum_{i=1}^{n} \alpha_i - C \sum_{i=1}^{n} \xi_i$$

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i k(x_i, x) + w_0$$
Kernel functions for SVM

An example of kernel that maps data to a feature space explicitly

\[ k(a, b) \triangleq (1 + a^T b)^2 = (1 + a_1 b_1 + a_2 b_2)^2 \]

\[ = 1 + 2a_1 b_1 + 2a_2 b_2 + a_1^2 b_1^2 + 2a_1 a_2 b_1 b_2 + a_2^2 b_2^2 \]

\[ = (1, \sqrt{2}a_1, \sqrt{2}a_2, a_1^2, \sqrt{2}a_1 a_2, a_2^2)(1, \sqrt{2}b_1, \sqrt{2}b_2, b_1^2, \sqrt{2}b_1 b_2, b_2^2)^T \]

\[ = \phi(a)^T \phi(b) \]

Popular kernels

<table>
<thead>
<tr>
<th>Kernel</th>
<th>( k(x_i, x_j) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial</td>
<td>((1 + \langle x_i, x_j \rangle)^d)</td>
</tr>
<tr>
<td>Radial basis function (RBF)</td>
<td>( e^{-\gamma |x_i - x_j|^2}, \ \gamma &gt; 0 )</td>
</tr>
<tr>
<td>Hyperbolic tangent</td>
<td>( \tanh(\kappa_1 \langle x_i, x_j \rangle + \kappa_2), \ \kappa_1 &gt; 0, \kappa_2 &lt; 0 )</td>
</tr>
</tbody>
</table>

where \( \langle x_i, x_j \rangle \) is an inner product (e.g. dot product) between \( x_i \) and \( x_j \).
Making kernels

How can we ensure if a kernel works as an inner product in a feature space?

It should satisfy:

- \( k(x, z) = \langle \phi(x), \phi(z) \rangle = \langle \phi(z), \phi(x) \rangle = k(z, x) \)
- \( k(x, z)^2 \leq k(x, x) k(z, z) \)
- \( K = (k(x_i, x_j)) \), which is a \( n \)-by-\( n \) matrix, is positive semi-definite.

Mercer’s theorem:

Suppose \( k \) is a continuous symmetric non-negative definite kernel, then \( k \) can be expressed as:

\[
k(x, z) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i(z)
\]

where \( \{\phi_i\} \) are eigen-functions, \( \|\phi_i\| = 1 \), and \( \{\lambda_i\} \) are positive eigenvalues \( \lambda_i > 0 \).
Making kernels from kernels

Letting $k_1$, $k_2$, and $k_3$ are kernels, we can create a new kernel $k$.

- $k(x, z) = k_1(x, z) + k_2(x, z)$
- $k(x, z) = ak_1(x, z)$, $a > 0$
- $k(x, z) = k_1(x, z) k_2(x, z)$
- $k(x, z) = f(x) f(z)$
- $k(x, z) = k_3(\phi(x), \phi(z))$
- $k(x, z) = x^T B z$, where $B$ is a $n$-by-$n$ matrix
Assuming the class $\mathcal{F}$ of real-valued functions on the ball of radius $R$ in $\mathbb{R}^n$ as

$$\mathcal{F} = \{ x \mapsto w \cdot x : \|w\| \leq 1, \|x\| \leq R \}.$$ 

If a classifier $\text{sgn}(f) \in \text{sgn}(\mathcal{F})$ has margin at least $\gamma$ on all the training examples, with probability at least $1 - \delta$ over $n$ random examples, $f$ has error no more than

$$L_D(f) \leq \frac{k}{n} + \sqrt{\frac{c}{n} \left( \frac{R^2}{\gamma^2} \log^2 n + \log \left( \frac{1}{\delta} \right) \right)},$$

where $k$ is the number of labelled training examples with margin less than $\gamma$, $c$ is a constant,

$$\text{VC-dim}(f) \leq \min \left( \frac{R^2}{\gamma^2}, n \right) + 1$$
Experiments on US Postal Service Database


US Postal Service Database (handwritten digits):

<table>
<thead>
<tr>
<th>Training samples</th>
<th>7300</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test samples</td>
<td>2000</td>
</tr>
<tr>
<td>Image resolution</td>
<td>16 × 16 pixels</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Err. [%]</th>
<th>Support vectors</th>
<th>Dimensionality of feature space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human performance</td>
<td>2.5</td>
<td>200</td>
<td>256</td>
</tr>
<tr>
<td>Decision tree, CART</td>
<td>17.0</td>
<td>127</td>
<td>～ 33000</td>
</tr>
<tr>
<td>Decision tree, V4.5</td>
<td>16.0</td>
<td>148</td>
<td>～ 1 × 10^6</td>
</tr>
<tr>
<td>Best 2 layer NN</td>
<td>6.6</td>
<td>165</td>
<td>～ 1 × 10^9</td>
</tr>
<tr>
<td>LeNet1 (5 layers)</td>
<td>5.1</td>
<td>175</td>
<td>～ 1 × 10^12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>185</td>
<td>～ 1 × 10^14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>190</td>
<td>～ 1 × 10^16</td>
</tr>
</tbody>
</table>

\[ d: \text{degree of polynomial kernel} \]
Some notes on SVMs

• How to find $w_0$? ... use $y_i(w^T x_i + w_0) = 1$ for support vectors
• How to choose the regulariser $C$? ... use a validation set
• How to solve the constrained quadratic optimisation problem in SVM practically? It requires a kernel matrix of $n$-by-$n$.
  ○ Gradient, sub-gradient, coordinate ascent/descent
  ○ Sequential Minimal Optimisation (SMO) [John Platt, 1998]
  ○ LIBSVM [Chih-Chung Chang and Chih-Jen Lin]: a SVM software tool with SMO

• How to apply SVMs to multi-class classification problems?
• Performance deterioration (NB: not very specific to SVMs)
  ○ Heavily-overlapped data sets
  ○ Imbalanced data sets
  ○ (Too many support vectors)
  ○ (Large data sets)
• Output interpretability
Consider a SVM with a linear kernel run on the following data set.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>4.0</td>
<td>1</td>
</tr>
<tr>
<td>4.0</td>
<td>2.0</td>
<td>1</td>
</tr>
<tr>
<td>4.0</td>
<td>4.0</td>
<td>1</td>
</tr>
<tr>
<td>0.0</td>
<td>2.0</td>
<td>2</td>
</tr>
<tr>
<td>2.0</td>
<td>-1.0</td>
<td>2</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>2</td>
</tr>
</tbody>
</table>

1. Using your intuition, what weight vector do you think will result from training an SVM on this data set?

2. Plot the data and the decision boundary of the weight vector you have chosen.

3. Which are the support vectors? What is the margin of this classifier?