SVM Active Learning

Soufyane El Allali

July 25, 2006
1. Support Vector Machines recap
2. Defining version space
3. Active learning
4. SVM active learning for image retrieval
5. References and further reading
Let us consider the following: \( \{x_1...x_n\} \) are vectors in some space \( X \subseteq \mathbb{R}^d \). And \( \{y_1...y_n\} \) are their corresponding labels, where \( y_i \in \{-1, 1\} \).

**SVMs** allow one to project the original training data in space \( X \) to a higher dimensional feature vector \( F \) via a **Mercer kernel** \( K \). Therefore we are considering the set of classifiers of the form:

\[
f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x)
\]

Given a kernel \( K \) satisfying **Mercer’s condition** \( K(u, v) = \Phi(u) \cdot \Phi(v) \) where \( \Phi : X \rightarrow F \), we can rewrite \( f \) as:

\[
f(x) = w \cdot \Phi(x), \text{ where } w = \sum_{i=1}^{n} \alpha_i \Phi(x_i).
\]
A simple linear SVM
One of the commonly used kernels is the radial basis function kernel:

\[ K(u, v) = e^{-\gamma(u-v) \cdot (u-v)} \]

For these kernels one property holds: \[ \| \Phi(x_i) \| = \lambda \] for some fixed \( \lambda \).

**Proof:**

\[ \| \Phi(x_i) \| = \sqrt{\Phi(x_i) \cdot \Phi(x_i)} = \sqrt{K(x_i, x_i)} = \sqrt{e^{-\gamma(x_i-x_i) \cdot (x_i-x_i)}} = \sqrt{e^0} = 1 \]
Let us consider the set of hypotheses $f$, that is the set of hyperplanes that separate the data in the feature space $F$.

$$H = \{ f \mid f(x) = \frac{w \cdot \Phi(x)}{\|w\|}, \text{where } w \in W \}$$

$W$ is the parameter space and is equal to $F$.

The version space $V$ is therefore defined as:

$$V = \{ f \in H \mid \forall i \in \{1..n\} : y_i f(x_i) > 0 \}$$

Since there is a bijection (an exact correspondence) between $w$ and $f \in H$. We redefine $V$ as:

$$V = \{ w \in W \mid \|w\| = 1, y_i (w \cdot \Phi(x_i)) > 0, i = 1..n \}$$
By definition points in $W$ correspond to hyperplanes in $F$. How about the converse?
Suppose we have a new **training instance** \( x_i \) with **label** \( y_i \). Then any **separating hyperplane must satisfy** \( y_i(w \cdot \Phi(x_i)) > 0 \).

**Rather than** viewing \( w \) as the **normal** of a hyperplane in \( F \), we think of \( y_i\Phi(x_i) \) as being the **normal** of a hyperplane in \( W \).

**Therefore** \( y_i(w \cdot \Phi(x_i)) = w \cdot y_i\Phi(x_i) > 0 \) defines a **half-space in** \( W \).

\( w \cdot y_i\Phi(x_i) = 0 \) defines a **hyperplane in** \( W \) which acts as one of the **boundaries to** version space \( V \).
Version space duality
Version space duality 2D
SVMs find the hyperplane that maximizes the margin in $F$.

$$\maximize_{w \in F} \min_i \{y_i(w \cdot \Phi(x_i))\}$$
subject to: $\|w\| = 1$ and $y_i(w \cdot \Phi(x_i)) > 0$ for $i = 1..n$

These conditions restrict the solution to lie in version space. Given the duality between $F$ and $W$ and since $\|\Phi(x_i)\| = 1$, then each $y_i\Phi(x_i)$ is a unit normal vector of a hyperplane in $W$ that delimits the version space.
We want to find the point in version space that maximizes the minimum distance to any of the hyperplanes. SVMs find the center of the largest radius hypersphere whose center is in version space and whose surface does not intersect with the hyperplanes corresponding to the labeled instances. The touched hyperplanes by this maximal radius hypersphere correspond to the support vectors and the radius of the hypersphere is the margin of the SVM.
An SVM classifier in version space
Let us first define the following:

- A pool of **unlabeled instances** $U$
- An **active learner** $l$ with three components $(f, q, X)$
- A **classifier** $f : X \rightarrow \{-1, 1\}$ trained on labeled data $X$, (and maybe on some of $U$)
- The **querying function** $q(X)$

The main difference between an active learner and a passive learner is the querying component $q$. How to choose the next unlabeled instance in the query-pool?
Definition: $\text{Area}(V)$ is the surface area that the version space $V$ occupies on the hypersphere $\| w \| = 1$.

In order to reduce the version space as fast as possible, we can intuitively choose a pool-query that halves the version space.
A **Simple** method is to learn an **SVM** based on the **existing labeled data** $X$ and **choose** as the **next** instance to **query** the pool **instance** that comes **closest** to the **hyperplane** in $F$. 
Simple margin method
SVM\textsubscript{Active} algorithm summary

- Learn an SVM on the current labeled data
- If this is the first feedback round, ask the user to label twenty randomly selected images.
- Otherwise, ask the user to label the twenty pool images closest to the SVM boundary.

After the relevance feedback rounds have been performed, \textit{SVM\textsubscript{Active}} retrieves the \textit{top} \textit{k} most relevant images:

- Learn a final SVM on the labeled data
- The final SVM boundary separates \textbf{relevant} images from \textbf{irrelevant} ones. Display the \textit{k} \textbf{relevant} images that are farthest from the SVM boundary.
Example

Initializing

Feedback Round 1
SVM Active Learning

SVM active learning for image retrieval

Feedback Round 2

Feedback Round 3
SVM Active Learning
SVM active learning for image retrieval

Fifth Screen of Results

Sixth Screen of Results
Demo


3. URL: [http://ai.stanford.edu/~stong/research.html](http://ai.stanford.edu/~stong/research.html)