Training algorithms for fuzzy support vector machines with noisy data

Presented by Josh Hoak

Chun-fu Lin[1] Sheng-de Wang[1]

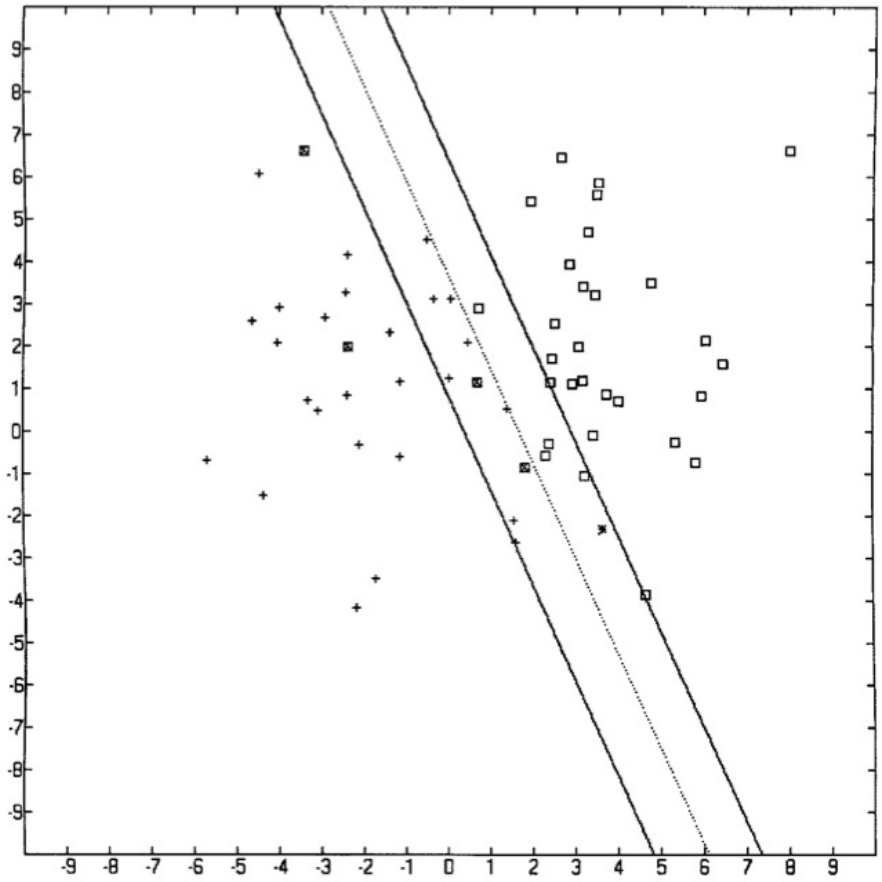
[1]National Taiwan University

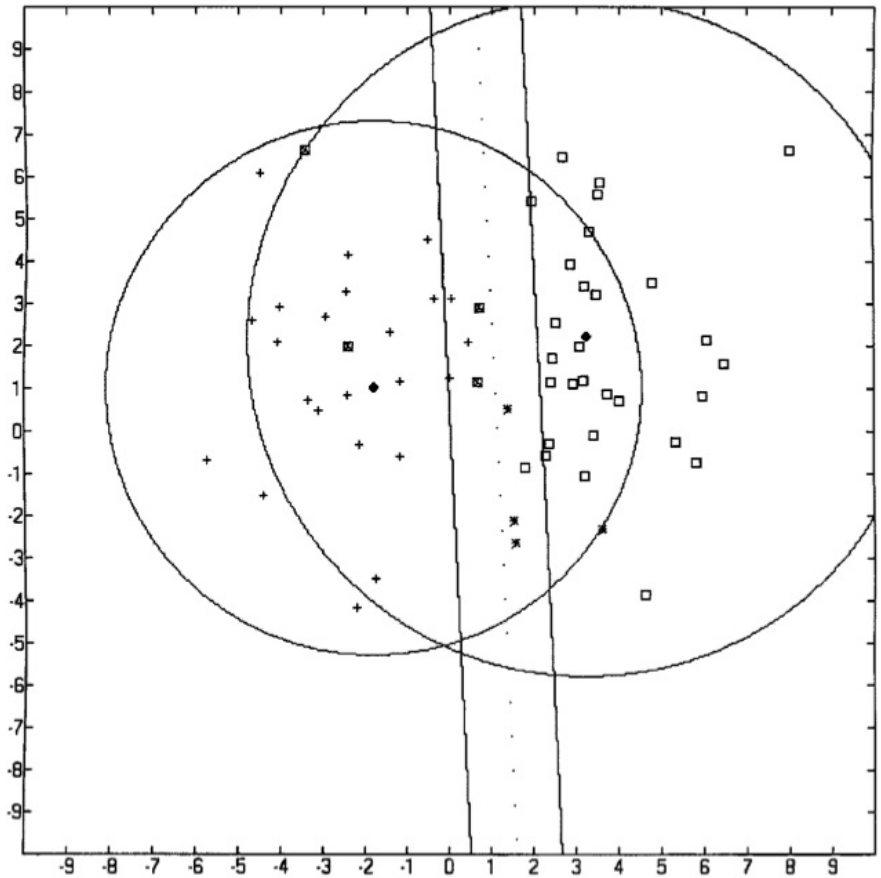
13 April 2010

Prelude

- **Problem:** SVMs are particularly susceptible to outliers. Fuzzy SVMs are a method to cope with outliers in SVMs.

A Motivating Example





Recap: Linear Support Vector Machine

- **Training Data:** $(y_1, \mathbf{x}_1) \ldots (y_m, \mathbf{x}_m) : x_i \in \mathbf{R}^n, y_i \in \{1, -1\}$
- **Goal:** Draw a hyperplane separating data with two classes; that is, find a vector $\mathbf{w}$ and a translation b such that

$$\mathbf{w} \cdot \mathbf{x} + b = 0,$$

where $\mathbf{x}$ is an example of from the training data, subject to the **constraint**:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \text{for } i = 1, \ldots l$$

Non-linear SVM

- **Non-linear Mapping:** Transform the data and then try to separate. Let $\phi : \mathbf{R}^n \longrightarrow \mathcal{F}$, where $\mathcal{F}$ indicates the feature space. Then,

$$\mathbf{z} = \phi(\mathbf{x})$$

- **Error term:** Add an error term ξ to the constraint:

$$y_i(\mathbf{w} \cdot \mathbf{z}_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, l$$

Non-linear SVMs continued

- Thus, we can minimize:

$$\frac{1}{2}\mathbf{w}\cdot\mathbf{w} - C\sum_{i=1}^{l}\xi_i,$$

subject to

$$y_i(\mathbf{w}\cdot\mathbf{z}_i - b) \geq 1 - \xi_i$$

Non-linear SVMs continued

- **Kernel Trick:** Find a *kernel* function $K(\cdot,\cdot) : \mathbf{R}^n \times \mathbf{R}^n \longrightarrow \mathcal{F}$ such that:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(x_i) \cdot \phi(x_j) = \mathbf{z}_i \cdot \mathbf{z}_j$$

- **Construct the optimal hyperplane:**

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{z}_i$$

- Optimal hyperplane is then:

$$f_H(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

Decision function: $f_D(\mathbf{x}) = \text{sign}(f_{\mathbf{H}}(\mathbf{X}))$

Fuzzy SVMs

- **Training Data**

$$(y_1, \mathbf{x}_1, s_1), \ldots, (y_l, \mathbf{x}_l, s_l), \quad \sigma \leq s_i \leq 1, \quad (\sigma > 0)$$

- **Fuzzy membership:** s_i is viewed as our *confidence* that the corresponding point $\mathbf{x}_i$ has class y_i

Fuzzy SVMs

- **Optimal Hyperplane:** The solution to minimizing:

$$\frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{i=1}^{l} s_i\xi_i,$$

constrained by

$$y_i(\mathbf{w}\cdot\mathbf{z}_i + b) \geq 1 - \xi_i$$

- Equivalently we can write:

$$\frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{I=1}^{l} \psi(\xi_i)$$

A more familiar formulation

- Maximize:

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i \mathbf{x}_j)$$

- Subject to:

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

$$0 \leq \alpha_i \leq s_i C, \quad i = 1, \ldots, l$$

How do we find the confidence values?

Fuzzy SVMs: The error function

Note: We model the theoretical error $\psi(\xi_i)$ by the probability that a point is noise – $p_x(\mathbf{x}_i)$. The error function becomes $\sum_{i=1}^{l} p_x(\mathbf{x}_i)\xi_i$

One model:

$$p_x(\mathbf{x}_i) = \begin{cases} 1, & \text{if } h(\mathbf{x}_i) > h_c, \\ \sigma, & \text{if } h(\mathbf{x}_i) < h_T, \\ \sigma + (1-\sigma)\left(\frac{h(\mathbf{x}_i)-h_T}{h_C-h_T}\right)^d, & \textit{otherwise}. \end{cases}$$

New definitions: h_c is the confidence factor, h_T is the 'trashy' factor, $h(x)$ is a heuristic function.

Fuzzy SVMs: The error function

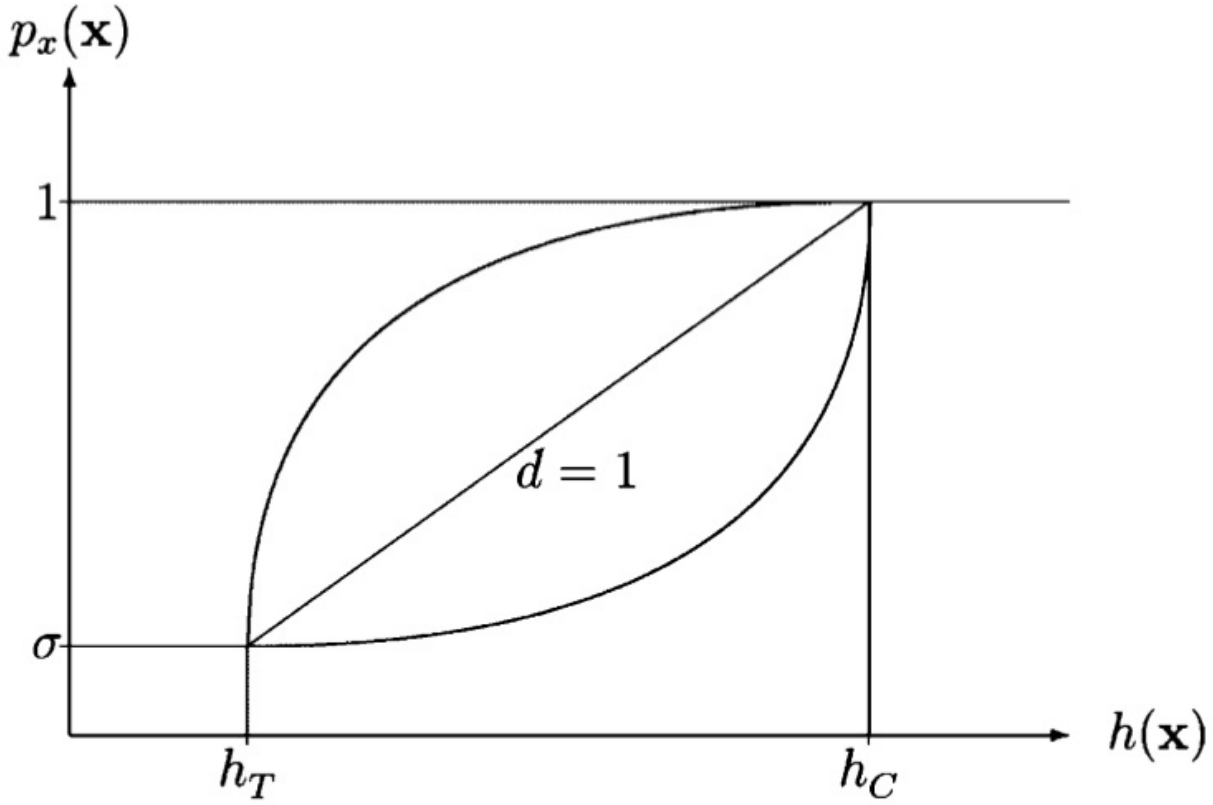


Fig. 1. The mapping between the probability density function $p_x(\boldsymbol{x})$ and the heuristic function $h(\boldsymbol{x})$.

Generating fuzzy memberships – $p_x(\mathbf{x})$

- We choose $\sigma > 0$ as a lower bound.
- Let's suppose that our fuzzy membership (or outlier status) is based on only one feature t. Then, we have:

$$s_i = h(\mathbf{x}_i) = f(t_i).$$

Generating fuzzy memberships (continued)

- Let the maximum of these be t_{max} and let t_{min} be the minimum; when $t_i = t_{min}$, we want the output to be σ.
- If we make s_i be a linear function of t then we get

$$s_i = f(t_i) = at_i + b$$

- Solving the system of equations:

$$\sigma = a(t_{min}) + b$$
$$1 = a(t_{max}) + b$$

Generating fuzzy memberships (continued)

- Solving, we get:

$$s_i = f(t_i) = \frac{1-\sigma}{t_{max} - t_{min}} t_i + \frac{t_{max}\sigma - t_{min}}{t_{max} - t_{min}}$$

- If we wish to make the function polynomial, we get:

$$s_i = f(t_i) = (1-\sigma)\left(\frac{t_i - t_{min}}{t_{max} - t_{min}}\right)^2 + \sigma$$

Fuzzy SVM: The heuristic function

Strategy 1: **Kernel-target alignment.** Defined as

$$A_{KT} = \frac{\sum_{i=1}^{l} f_K(\mathbf{x}_i, y_i)}{l\sqrt{\sum_{i,j=1}^{l} K^2(\mathbf{x}_i, \mathbf{x}_j)}}$$

where $f_K(\mathbf{x}_i, y_i) = \sum_{j=1}^{l} y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$

Idea: Use $f_K(\mathbf{x}_i, y_i)$ as the heuristic function.

Fuzzy SVMs: The heuristic function—

Strategy 2: **k-NN.** Find the nearest neighbors of a data point $\mathbf{x}_i$ (of the same class). Assume that the data point with fewer nearest neighbors (of the same class) has higher probability of being noisy data. Let the heuristic function be:

$$h(\mathbf{x}_i) = n_i,$$

where n_i is the number of nearest neighbors.

Overall Procedure

1. Use the original algorithm of SVMs to get the optimal kernel parameters and the regularization parameter C.
2. Fix the kernel parameters and the regularization parameter C from (1), and then find the other parameters in FSVMs.
   - 2.1 Define the heuristic function.
   - 2.2 Use exhaustive search to find the h_T, h_C, d, and σ.

Results

Table 1: Error rates for SVMs and FSVMs using KT and k-NN

	TR	SVMs	KT	k-NN
Banana	6.7	11.5 $\pm$ 0.7	*10.4 $\pm$ 0.5	11.4 $\pm$ 0.6
B. Cancer	18.3	26.0 $\pm$ 4.7	25.3 $\pm$ 4.4	*25.2 $\pm$ 4.1
Diabetes	19.4	23.5 $\pm$ 1.7	*23.3 $\pm$ 1.7	23.5 $\pm$ 1.7
German	16.2	23.6 $\pm$ 2.1	*23.3 $\pm$ 2.3	23.6 $\pm$ 2.1
Heart	12.8	16.0 $\pm$ 3.3	*14.2 $\pm$ 3.1	23.6 $\pm$ 2.1
Image	3.0	3.0 $\pm$ 0.6	*2.9 $\pm$ 0.7	-
Ringnorm	0.0	*1.7 $\pm$ 0.1	-	-
F. Solar	32.6	*32.4 $\pm$ 1.8	32.4 $\pm$ 1.8	32.4 $\pm$ 1.8
Splice	0.0	*10.9 $\pm$ 0.7	-	-
Thyroid	0.4	4.8 $\pm$ 2.2	*4.$\pm$ 2.3	-
Titanic	19.6	22.4 $\pm$ 1.0	*22.3$\pm$0.9	*22.3 $\pm$ 1.1
Twonorm	0.4	3.0 $\pm$ 0.2	*2.4 $\pm$ 0.1	2.9 $\pm$ 0.2
Waveform	3.5	*9.9 $\pm$ 0.4	9.9 $\pm$ 0.4	-

13 data sets from the UCI, DELVE and STAT- LOG

Multi-class SVM

Ensemble Method Use the OAA (One Against All) method of multiple-classification. Given M classes, we construct M binary SVM classifiers that separate one class from the rest. The class associated with the classifier that outputs the highest value given an example is then chosen.

Fuzzy Membership Function: When training,

$$Fuzz(\mathbf{x}) = \begin{cases} 1 & \text{if the output of the ensemble on } \mathbf{x} \text{ is } 1, \\ h & \text{if the output of ensemble on } \mathbf{x} \text{ is } -1. \end{cases}$$

Confusion Matrix

Actual Class	Predicted Class				
	Earn	Acq	Money-fx	Grain	Total
Earn	679	3	3	7	692
Acq	24	425	2	2	453
Money-Fx	6	5	86	4	101
Grain	12	1	2	87	102
Total	721	434	93	100	1348

Note: Overall Accuracy = 0.945

Data: Reuters Documents

Results

Macro-average perf. of OAA-SVM and OAA-FSVM

Classifier	4-fold		
	Precision	Recall	F Measure
OAA-FSVM(1, 0.5)	**0.775**	0.624	0.672
OAA-FSVM(1, 0.6)	0.772	**0.630**	**0.675**
OAA-SVM	0.764	0.616	0.659

Thoughts

- Statistically significant results?
- Diminishing returns?
- Cost?
- How much *do* outliers affect the model?