Introduction to Support Vector Machines

Hsuan-Tien Lin

Learning Systems Group, California Institute of Technology

Talk in NTU EE/CS Speech Lab, November 16, 2005

Setup

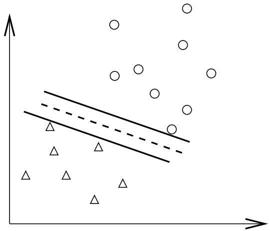
- fixed-length example: D-dimensional vector x, each component is a feature
  - raw digital sampling of a 0.5 sec. wave file
  - DFT of the raw sampling
  - a fixed-length feature vector extracted from the wave file
- label: a number $y \in \mathcal{Y}$
  - binary classification: is there a man speaking in the wave file? ($y = +1$ if man, $y = -1$ if not)
  - multi-class classification: which speaker is speaking? ($y \in \{1, 2, \cdots, K\}$)
  - regression: how excited is the speaker? ($y \in \mathbb{R}$)

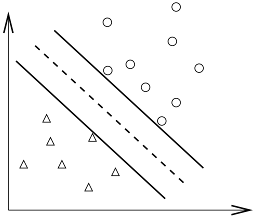
Binary Classification Problem

- learning problem: given training examples and labels $\{(x_i, y_i)\}_{i=1}^N$, find a function $g(x) : \mathcal{X} \to \mathcal{Y}$ that predicts the label of unseen x well
  - vowel identification: given training wave files and their vowel labels, find a function $g(x)$ that translates wave files to vowel well
- we will focus on binary classification problem: $\mathcal{Y} = \{+1, -1\}$
- most basic learning problem, but very useful and can be extended to other problems
- illustrative demo: for examples with two different color in a D-dimensional space, how can we "separate" the examples?

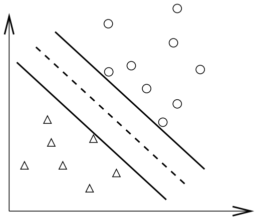
Hyperplane Classifier

- use a hyperplane to separate the two colors: $g(x) = \text{sign}\left(w^T x + b\right)$
- if $w^T + b \geq 0$, the classifier returns $+1$, otherwise the classifier returns -1
- possibly lots of hyperplanes satisfying our needs, which one should we choose?





SVM: Large-Margin Hyperplane Classifier



- margin $\rho_i = y_i(w^T x + b)/\|w\|_2$:
  - does y_i agree with $w^T x + b$ in sign?
  - how large is the distance between the example and the separating hyperplane?
- large positive margin $\rightarrow$ clear separation $\rightarrow$ low risk classification
- idea of SVM: maximize the minimum margin

$$\max_{w,b} \quad \min_i \rho_i$$
$$\text{s.t.} \quad \rho_i = y_i(w^T x_i + b)/\|w\|_2 \geq 0$$

Hard-Margin Linear SVM

- maximize the minimum margin

$$\begin{aligned} \max_{w,b} \quad & \min_i \rho_i \\ \text{s.t.} \quad & \rho_i = y_i(w^T x_i + b)/\|w\|_2 \geq 0, i = 1, \ldots, N. \end{aligned}$$

- equivalent to

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} w^T w \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1, i = 1, \ldots, N. \end{aligned}$$

– hard-margin linear SVM

- quadratic programming with $D + 1$ variables: well-studied in optimization
- is the hard-margin linear SVM good enough?

Soft-Margin Linear SVM

- hard-margin – hard constraints on separation:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} w^T w \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1, i = 1, \ldots, N. \end{aligned}$$

- no feasible solution if some noisy outliers exist
- soft-margin – soft constraints as cost:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} w^T w + C \sum_i \xi_i \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, \ldots, N. \end{aligned}$$

- allow the noisy examples to have $\xi_i > 0$ with a cost
- is linear SVM good enough?

Soft-Margin Nonlinear SVM

- what if we want a boundary $g(x) = \text{sign}\left(x^T x - 1\right)$?
- can never be constructed with a hyperplane classifier $\text{sign}\left(w^T x + b\right)$
- however, we can have more complex feature transforms:

$$\phi(x) = [(x)_1, (x)_2, \cdots, (x)_D, (x)_1(x)_1, (x)_1(x)_2, \cdots, (x)_D(x)_D]$$

- there is a classifier $\text{sign}\left(w^T \phi(x) + b\right)$ that describes the boundary
- soft-margin nonlinear SVM:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} w^T w + C \sum_i \xi_i \\ \text{s.t.} \quad & y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, \ldots, N. \end{aligned}$$

– with nonlinear $\phi(\cdot)$

Feature Transformation

- what feature transforms $\phi(\cdot)$ should we use?
- we can only extract finite small number of features, but we can use unlimited number of feature transforms
- traditionally:
  - use domain knowledge to do feature transformation
  - use only "useful" feature transformation
  - use a small number of feature transformation
- control the goodness of fitting by suitable choice of feature transformation
- what if we use "infinite number" of feature transformation, and let the algorithm decide a good w automatically?
  - would infinite number of transformations introduce overfitting?
  - are we able to solve the optimization problem?

Dual Problem

- infinite quadratic programming if infinite $\phi(\cdot)$:

$$\begin{aligned}\min_{w,b} \quad & \frac{1}{2}w^T w + C\sum_i \xi_i \\ \text{s.t.} \quad & y_i(w^T\phi(x_i)+b) \geq 1-\xi_i, \\ & \xi_i \geq 0, i = 1,\ldots,N.\end{aligned}$$

- luckily, we can solve its associated dual problem:

$$\begin{aligned}\min_{\alpha} \quad & \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ \text{s.t.} \quad & y^T\alpha = 0, \\ & 0 \leq \alpha_i \leq C, \\ & Q_{ij} \equiv y_i y_j \phi^T(x_i)\phi(x_j)\end{aligned}$$

- α: N-dimensional vector

Solution of the Dual Problem

- associated dual problem:

$$
\begin{aligned}
\min_{\alpha} \quad & \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\
\text{s.t.} \quad & y^T\alpha = 0, \\
& 0 \le \alpha_i \le C, \\
& Q_{ij} \equiv y_i y_j \phi^T(x_i)\phi(x_j)
\end{aligned}
$$

- equivalent solution:

$$
g(x) = \text{sign}\left(w^T x + b\right) = \text{sign}\left(\sum y_i \alpha_i \phi^T(x_i)\phi(x) + b\right)
$$

- no need for w and $\phi(x)$ explicitly if we can compute $K(x, x') = \phi^T(x)\phi(x')$ efficiently

Kernel Trick

- let kernel $K(x, x') = \phi^T(x)\phi(x')$
- revisit: can we compute the kernel of

$$\phi(x) = [(x)_1, (x)_2, \cdots, (x)_D, (x)_1(x)_1, (x)_1(x)_2, \cdots, (x)_D(x)_D]$$

  efficiently?
- well, not really
- how about this?

$$\phi(x) = \left[\sqrt{2}(x)_1, \sqrt{2}(x)_2, \cdots, \sqrt{2}(x)_D, (x)_1(x)_1, \cdots, (x)_D(x)_D\right]$$

- $K(x, x') = (1 + x^T x')^2 - 1$

Different Kernels

- types of kernels
  - linear $K(x, x') = x^T x'$,
  - polynomial: $K(x, x') = (ax^T x' + r)^d$
  - Gaussian RBF: $K(x, x') = \exp(-\gamma \|x - x'\|_2^2)$
  - Laplacian RBF: $K(x, x') = \exp(-\gamma \|x - x'\|_1)$
- the last two equivalently have feature transformation in **infinite** dimensional space!
- new paradigm for machine learning: use many many feature transformations, control the goodness of fitting by large-margin (clear separation) and violation cost (amount of outlier allowed)

Support Vectors: Meaningful Representation

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ \text{s.t.} \quad & y^T\alpha = 0, \\ & 0 \le \alpha_i \le C, \end{aligned}$$

- equivalent solution:

$$g(x) = \text{sign}\left(\sum y_i\alpha_i K(x_i, x) + b\right)$$

- only those with $\alpha_i > 0$ are needed for classification – support vectors
- from optimality conditions, α_i:
  - "$= 0$": no need in constructing the decision function, away from the boundary or on the boundary
  - "> 0 and $< C$": free support vector, on the boundary
  - "$= C$": bounded support vector, violate the boundary ($\xi_i > 0$) or on the boundary

Why is SVM Successful?

- infinite number of feature transformation: suitable for conquering nonlinear classification tasks
- large-margin concept: theoretically promising
- soft-margin trade-off: controls regularization well
- convex optimization problems: possible for good optimization algorithms (compared to Neural Networks and some other learning algorithms)
- support vectors: useful in data analysis and interpretation

Why is SVM Not Successful?

- SVM can be sensitive to scaling and parameters
- standard SVM is only a “discriminative” classification algorithm
- SVM training can be time-consuming when N is large and the solver is not carefully implemented
- infinite number of feature transformation $\Leftrightarrow$ mysterious classifier

Useful Extensions of SVM

- multiclass SVM: use 1vs1 approach to combine binary SVM to multiclass
  – the label that gets more votes from the classifiers is the prediction
- probability output: transform the raw output $w^T\phi(x) + b$ to a value between $[0, 1]$ to mean $P(+1|x)$
  – use a sigmoid function to transform from $\mathbb{R} \to [0, 1]$
- infinite ensemble learning (Lin and Li 2005):
  if the kernel $K(x, x') = -\|x - x'\|_1$ is used for standard SVM, the classifier is equivalently

$$g(x) = \text{sign}\left(\int w_\theta s_\theta(x) d\theta + b\right)$$

where $s_\theta(x)$ is a thresholding rule on one feature of x.

Basic Use of SVM

- scale each feature of your data to a suitable range (say, $[-1, 1]$)
- use a Gaussian RBF kernel $K(x, x') = \exp(-\gamma\|x - x'\|_2^2)$
- use cross validation and grid search to determine a good (γ, C) pair
- use the best (γ, C) on your training set
- do testing with the SVM classifier

all included in LIBSVM (from Lab of Prof. Chih-Jen Lin)

Advanced Use of SVM

- include domain knowledge by specific kernel design (e.g. train a generative model for feature extraction, and use the extracted feature in SVM to get discriminative power)
- combining SVM with your favorite tools (e.g. HMM + SVM for speech recognition)
- fine-tune SVM parameters with specific knowledge of your problem (e.g. different costs for different examples?)
- interpreting the SVM results you get (e.g. are the SVs meaningful?)

Resources

- LIBSVM: `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
- LIBSVM Tools:
  `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools`
- Kernel Machines Forum: `http://www.kernel-machines.org`
- Hsu, Chang, and Lin: A Practical Guide to Support Vector Classification
- my email: `htlin@caltech.edu`

acknowledgment: some figures obtained from Prof. Chih-Jen Lin