SVM and Its Applications to Text Classification

Dr. Tie-Yan Liu
WSM Group, MSR Asia
2006.3.23
Outline

• A Brief History of SVM
• SVM: A Large-Margin Classifier
  – Linear SVM
  – Kernel Trick
  – Fast implementation: SMO
• SVM for Text Classification
  – Multi-class Classification
  – Multi-label Classification
  – Our Hierarchical Classification Tool
History of SVM

- SVM is inspired from statistical learning theory [3]
- SVM was first introduced in 1992 [1]
- SVM becomes popular because of its success in handwritten digit recognition
  - 1.1% test error rate for SVM. This is the same as the error rates of a carefully constructed neural network, LeNet 4.
    - See Section 5.11 in [2] or the discussion in [3] for details
- SVM is now regarded as an important example of “kernel methods”, arguably the hottest area in machine learning

What is a Good Decision Boundary?

• Consider a two-class, linearly separable classification problem

• Many decision boundaries!
  – The Perceptron algorithm can be used to find such a boundary

• Are all decision boundaries equally good?
Examples of Bad Decision Boundaries
Large-margin Decision Boundary

- The decision boundary should be as far away from the data of both classes as possible
  - We should maximize the margin, $m$

\[
m = \frac{2}{||w||}
\]
Finding the Decision Boundary

• Let \( \{x_1, \ldots, x_n\} \) be our data set and let \( y_i \in \{1,-1\} \) be the class label of \( x_i \)

• The decision boundary should classify all points correctly

\[ y_i(w^T x_i + b) \geq 1, \quad \forall i \]

• The decision boundary can be found by solving the following constrained optimization problem

\[
\text{Minimize} \quad \frac{1}{2} \|w\|^2 \\
\text{subject to} \quad y_i(w^T x_i + b) \geq 1 \quad \forall i
\]

• The Lagrangian of this optimization problem is

\[
\mathcal{L} = \frac{1}{2} \|w\|^2 - \sum_i \alpha_i \left( y_i(w^T x_i + b) - 1 \right) \quad \alpha_i \geq 0 \quad \forall i
\]
The Dual Problem

• By setting the derivative of the Lagrangian to be zero, the optimization problem can be written in terms of \( \alpha_i \) (the dual problem)

\[
\begin{align*}
\max W(\alpha) &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \\
\text{subject to } &\alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0
\end{align*}
\]

• This is a quadratic programming (QP) problem
  – A global maximum of \( \alpha_i \) can always be found

• \( w \) can be recovered by \( w = \sum_{i=1}^{n} \alpha_i y_i x_i \)

If the number of training examples is large, SVM training will be very slow because the number of parameters Alpha is very large in the dual problem.
KTT Condition

- The QP problem is solved when for all i,

\[ \alpha_i = 0 \quad \Rightarrow \quad y_i f(x_i) \geq 1 \]
\[ 0 < \alpha_i < C \quad \Rightarrow \quad y_i f(x_i) = 1 \]
\[ \alpha_i = C \quad \Rightarrow \quad y_i f(x_i) \leq 1 \]
Characteristics of the Solution

• KTT condition indicates many of the $\alpha_i$ are zero
  – $w$ is a linear combination of a small number of data points
• $x_i$ with non-zero $\alpha_i$ are called support vectors (SV)
  – The decision boundary is determined only by the SV
  – Let $t_j \ (j=1, \ldots, s)$ be the indices of the $s$ support vectors. We can write $w = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} x_{t_j}$

• For testing with a new data $z$
  – Compute $w^T z + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} (x_{t_j}^T z) + b$
    and classify $z$ as class 1 if the sum is positive, and class 2 otherwise.
A Geometrical Interpretation

\[ w^T x + b = 1 \]

\[ w^T x + b = -1 \]
Non-linearly Separable Problems

• We allow “error” $\xi_i$ in classification
Soft Margin Hyperplane

- By minimizing $\sum_i \xi_i$, $\xi_i$ can be obtained by
  \[
  \begin{align*}
  w^T x_i + b &\geq 1 - \xi_i & y_i = 1 \\
  w^T x_i + b &\leq -1 + \xi_i & y_i = -1 \\
  \xi_i &\geq 0 & \forall i
  \end{align*}
  \]
  $\xi_i$ are “slack variables” in optimization; $\xi_i=0$ if there is no error for $x_i$, and $\xi_i$ is an upper bound of the number of errors

- We want to minimize $\frac{1}{2}||w||^2 + C \sum_i \xi_i$
  
  $C$ : tradeoff parameter between error and margin

- The optimization problem becomes
  
  Minimize $\frac{1}{2}||w||^2 + C \sum_{i=1}^{n} \xi_i$
  subject to $y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$
The Optimization Problem

• The dual of the problem is

\[ \max \ W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \]

subject to \( C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0 \)

• \( \mathbf{w} \) is recovered as \( \mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j} \)

• This is very similar to the optimization problem in the linear separable case, except that there is an upper bound \( C \) on \( \alpha_i \) now

• Once again, a QP solver can be used to find \( \alpha_i \)
Extension to Non-linear Decision Boundary

• So far, we only consider large-margin classifier with a linear decision boundary, how to generalize it to become nonlinear?

• Key idea: transform \( \mathbf{x}_i \) to a higher dimensional space to “make life easier”
  – Input space: the space the point \( \mathbf{x}_i \) are located
  – Feature space: the space of \( \phi(\mathbf{x}_i) \) after transformation

• Why transform?
  – Linear operation in the feature space is equivalent to non-linear operation in input space
  – Classification can become easier with a proper transformation. In the XOR problem, for example, adding a new feature of \( x_1x_2 \) make the problem linearly separable
Transforming the Data

- Computation in the feature space can be costly because it is high dimensional
  - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue
The Kernel Trick

• Recall the SVM optimization problem

\[
\max \ W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j
\]

subject to \( C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0 \)

• The data points only appear as inner product

• As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly

• Many common geometric operations (angles, distances) can be expressed by inner products

• Define the kernel function \( K \) by \( K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \)
An Example for $\phi(.)$ and $K(.,.)$

- Suppose $\phi(.)$ is given as follows
  \[
  \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)
  \]

- An inner product in the feature space is
  \[
  \langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right)\rangle = (1 + x_1y_1 + x_2y_2)^2
  \]

- So, if we define the kernel function as follows, there is no need to carry out $\phi(.)$ explicitly
  \[
  K(x, y) = (1 + x_1y_1 + x_2y_2)^2
  \]

- This use of kernel function to avoid carrying out $\phi(.)$ explicitly is known as the kernel trick
Kernel Functions

• In practical use of SVM, only the kernel function (and not \( \phi(\cdot) \)) is specified
• Kernel function can be thought of as a similarity measure between the input objects
• Not all similarity measure can be used as kernel function, however Mercer's condition states that any positive semi-definite kernel \( K(x, y) \), i.e.

\[
\sum_{i,j} K(x_i, x_j)c_i c_j \geq 0
\]

can be expressed as a dot product in a high dimensional space.
Examples of Kernel Functions

• Polynomial kernel with degree $d$
  \[ K(x, y) = (x^T y + 1)^d \]

• Radial basis function kernel with width $\sigma$
  \[ K(x, y) = \exp(-||x - y||^2/(2\sigma^2)) \]
  – Closely related to radial basis function neural networks

• Sigmoid with parameter $\kappa$ and $\theta$
  \[ K(x, y) = \tanh(\kappa x^T y + \theta) \]
  – It does not satisfy the Mercer condition on all $\kappa$ and $\theta$
Modification Due to Kernel Function

• Change all inner products to kernel functions
• For training,

Original

$$\max \ W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

subject to $C \geq \alpha_i \geq 0$, $\sum_{i=1}^{n} \alpha_i y_i = 0$

With kernel function

$$\max \ W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

subject to $C \geq \alpha_i \geq 0$, $\sum_{i=1}^{n} \alpha_i y_i = 0$
Modification Due to Kernel Function

• For testing, the new data $z$ is classified as class 1 if $f \geq 0$, and as class 2 if $f < 0$

Original

$$w = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} x_{t_j}$$

$$f = w^T z + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} x_{t_j}^T z + b$$

With kernel function

$$w = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \phi(x_{t_j})$$

$$f = \langle w, \phi(z) \rangle + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} K(x_{t_j}, z) + b$$
Why SVM Works?

• The feature space is often very high dimensional. Why don’t we have the curse of dimensionality?
  – A classifier in a high-dimensional space has many parameters and is hard to estimate

• Vapnik argues that the fundamental problem is not the number of parameters to be estimated. Rather, the problem is about the flexibility of a classifier

• Typically, a classifier with many parameters is very flexible, but there are also exceptions
  – Let \( x_i = 10^i \) where \( i \) ranges from 1 to \( n \). The classifier \( y = \text{sign}\left(\sin(\alpha x)\right) \) can classify all \( x_i \) correctly for all possible combination of class labels on \( x_i \)
  – This 1-parameter classifier is very flexible
Why SVM Works?

• Vapnik argues that the flexibility of a classifier should not be characterized by the number of parameters, but by the capacity of a classifier
  – This is formalized by the “VC-dimension” of a classifier
• The addition of $\frac{1}{2}||w||^2$ has the effect of restricting the VC-dimension of the classifier in the feature space
• The SVM objective can also be justified by structural risk minimization: the empirical risk (training error), plus a term related to the generalization ability of the classifier, is minimized
• Another view: the SVM loss function is analogous to ridge regression. The term $\frac{1}{2}||w||^2$ “shrinks” the parameters towards zero to avoid overfitting
Choosing the Kernel Function

• Probably the most tricky part of using SVM.
• The kernel function is important because it creates the kernel matrix, which summarize all the data
• Many principles have been proposed (diffusion kernel, Fisher kernel, string kernel, ...)
• There are even research to estimate the kernel matrix from available information

• In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try for most applications.
• It was said that for text classification, linear kernel is the best choice, because of the already-high-enough feature dimension
Strengths and Weaknesses of SVM

• Strengths
  – Training is relatively easy
    • No local optimal, unlike in neural networks
  – It scales relatively well to high dimensional data
  – Tradeoff between classifier complexity and error can be controlled explicitly
  – Non-traditional data like strings and trees can be used as input to SVM, instead of feature vectors
  – By performing logistic regression (Sigmoid) on the SVM output of a set of data can map SVM output to probabilities.

• Weaknesses
  – Need to choose a “good” kernel function.
Summary: Steps for Classification

- Prepare the pattern matrix
- Select the kernel function to use
- Select the parameter of the kernel function and the value of C
  - You can use the values suggested by the SVM software, or you can set apart a validation set to determine the values of the parameter
- Execute the training algorithm and obtain the $\alpha_i$
- Unseen data can be classified using the $\alpha_i$ and the support vectors
Fast SVM Implementations

- SMO: Sequential Minimal Optimization
- SVM-Light
- LibSVM
- BSVM
- ......
SMO: Sequential Minimal Optimization

• Key idea
  – Divide the large QP problem of SVM into a series of smallest possible QP problems, which can be solved analytically and thus avoids using a time-consuming numerical QP in the loop (a kind of SQP method).
  – Space complexity: $O(n)$.
  – Since QP is greatly simplified, most time-consuming part of SMO is the evaluation of decision function, therefore it is very fast for linear SVM and sparse data.
SMO

- At each step, SMO chooses 2 Lagrange multipliers to jointly optimize, find the optimal values for these multipliers and updates the SVM to reflect the new optimal values.
- Three components
  - An analytic method to solve for the two Lagrange multipliers
  - A heuristic for choosing which multipliers to optimize
  - A method for computing $b$ at each step, so that the KTT conditions are fulfilled for both the two examples
Choosing Which Multipliers to Optimize

• First multiplier
  – Iterate over the entire training set, and find an example that violates the KTT condition.

• Second multiplier
  – Maximize the size of step taken during joint optimization.
  – $|E_1 - E_2|$, where $E_i$ is the error on the $i$-th example.
SVM for Text Classification
Text Categorization

• Typical features
  – Term frequency
  – Inverse document frequency

\[ IDF(w_i) = \log \left( \frac{n}{DF(w_i)} \right) \]

• TC is a typical multi-class multi-label classification problem.
  – SVM, with some additional heuristic, has been regarded as one of the best classification scheme for text data, based on many benchmark evaluations.

• TC is a high-dimensional sparse problem
  – SMO is a very good choice in this case.
Multi-Class SVM Classification

• 1-vs-rest
• 1-vs-1
  – MaxWin
  – DB2
  – Error Correcting Output Coding
• K-class SVM
1-vs-rest

- For any class $C$, train a binary classifier to distinguish $C$ from $\bar{C}$.
- For an unseen sample, find the binary classifier with highest confidence score for the final decision.
1-vs-1

- Train $C_N^2$ classifiers, which distinguish one class from another one.
  - Pairwise:
    - MaxWin ($C_N^2$ tests)
    - Error-correcting output code
  - DAG:
    - Pachinko-machine (N tests)
Error Correcting Output Coding

- Code Matrix ($M^{N\times K}$)
  - $N$ classes, $K$ classifiers
- Hamming Distance
- Class $C_i$ with Minimum Error wins

\[
\begin{array}{c|cccccccc}
M & 1:2 & 1:3 & 1:4 & 2:3 & 2:4 & 3:4 \\
\hline
1 & 1 & 1 & 1 & 0 & 0 & 0 \\
2 & -1 & 0 & 0 & 1 & 1 & 0 \\
3 & 0 & -1 & 0 & -1 & 0 & 1 \\
4 & 0 & 0 & -1 & 0 & -1 & -1 \\
\end{array}
\]

\[
\begin{array}{c|cccccccc}
M & 1,2 & 1,3 & 1,4 & 2,3 & 2,4 & 3,4 \\
\hline
1 & 1 & 1 & 1 & -1 & -1 & -1 \\
2 & 1 & -1 & -1 & 1 & 1 & -1 \\
3 & -1 & 1 & -1 & -1 & 1 & 1 \\
4 & -1 & -1 & 1 & -1 & 1 & 1 \\
\end{array}
\]
Intransitivity of DAG

- For $C_1, C_2, C_3$, if $C_3 >_d C_2, C_2 >_d C_1$, then $C_3 >_d C_1$, we say $>_d$ is transitive.
Divided-by-2 (DB2)

- Hierarchically divide the data into two subsets until every subset consists of only one class.
Divided-by-2 (DB2)

• Data partitioning criterion:
  – group the classes such that the resulting subsets have the largest margin.

• Trade-off: use clustering methods
  – k-mean: use the mean of each class
  – Balanced subsets: minimal difference in sample number.
K-class SVM

• Change the loss function and constraints

\[
\begin{align*}
\text{minimise} & \quad \phi(w, \xi) = \frac{1}{2} \sum_{m=1}^{k} (w_m \cdot w_m) + C \sum_{i=1}^{\ell} \sum_{m \neq y_i} \xi_i^m \\
\text{with constraints} & \quad (w_{y_i} \cdot x_i) + b_{y_i} \geq (w_m \cdot x_i) + b_m + 2 - \xi_i^m, \\
& \quad \xi_i^m \geq 0, \quad i = 1, \ldots, \ell \quad m \in \{1, \ldots, k\} \setminus y_i. 
\end{align*}
\]
Multi-label SVM Classification

- How does multi-label come?
  - Whole-vs-part
  - Share concepts
Whole-vs-part

• Common for parent-child relationship
  – Add an “Other” category, and do binary classification to distinguish the child from the other category.
  – Since the classification boundary is non-linear, kernel methods may be more effective.
Share concepts: Training

- **Mode-S**
  - Label multi-label data with the class to which the data most likely belonged, by some perhaps subjective criterion.

- **Mode-N**
  - consider the multi-label data as a new class

- **Mode-X**
  - Use the multi-label data more than once, using each example as a positive example of each of the classes to which it belongs.
Share concepts: Test

• P-cut
  – Label input testing data by all of the classes corresponding to positive SVM scores. If no scores are positive, label that data to the class with top score.

• S-cut
  – Train a threshold for each class by cross validation, and Label input testing data by all of the classes corresponding to higher scores than the threshold.

• R-cut
  – For any given test instance, always assign it \( r \) labels according to the decedent confidence scores.
  – \( r \) can be learned from training data.
Evaluation Criteria

• Micro-F1:  \[ MicroF1 = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} (TP_i + FP_i)} \]
  – Measure the overall classification accuracy (more consistent with the practical application scenario)

• Macro-F1:  \[ MacroF1 = \frac{1}{n} \sum_{i=1}^{n} \frac{TP_i}{TP_i + FP_i} \]
  – Measure the classification accuracy on the category level. Can reflect the classifier’s capability of dealing with rare categories.
References

- Martin Law, A Simple Introduction to Support Vector Machines.
Thanks

tyliu@microsoft.com
http://research.microsoft.com/users/tyliu