SUPPORT VECTOR MACHINE PADA INFORMATION RETRIEVAL

Oleh
I Ketut Purnamawan
Jurusan Manajemen Informatika Fakultas Teknik dan Kejuruan
Universitas Pendidikan Ganesha
tutpurna@yahoo.com

ABSTRAK

SVM merupakan *classsifier* yang mempunyai keunggulan dapat mengolah data berdimensi tinggi, tanpa mengalami penurunan performa yang signifikan. SVM sekarang ini semakin banyak dipergunakan. Pada *information retrieval*, SVM juga sudah banyak digunakan, khususnya pada bagian proses klasifikasi. Kemampuan SVM untuk mengolah data berdimensi besar sangat cocok untuk diterapkan pada data teks yang cenderung berdimensi besar. Pada tulisan ini dibahas perbandingan performa SVM dengan performa *classsifier* lain pada *information retrieval*. Pembahasan dilakukan berdasarkan hasil penelitian beberapa peneliti. Pada satu penelitian didapatkan bahwa SVM mengungguli *classsifier* lain. Pada penelitian lain didapatkan performa SVM berimbang dengan *classsifier* lain. Pada akhir tulisan dibahas mengenai langkah-langkah klasifikasi teks menggunakan SVM. Proses klasifikasi teks ini digunakan pada *information retrieval* data teks.
Kata kunci : SVM, *Information retrieval*, *classsifier.*

ABSTRACT

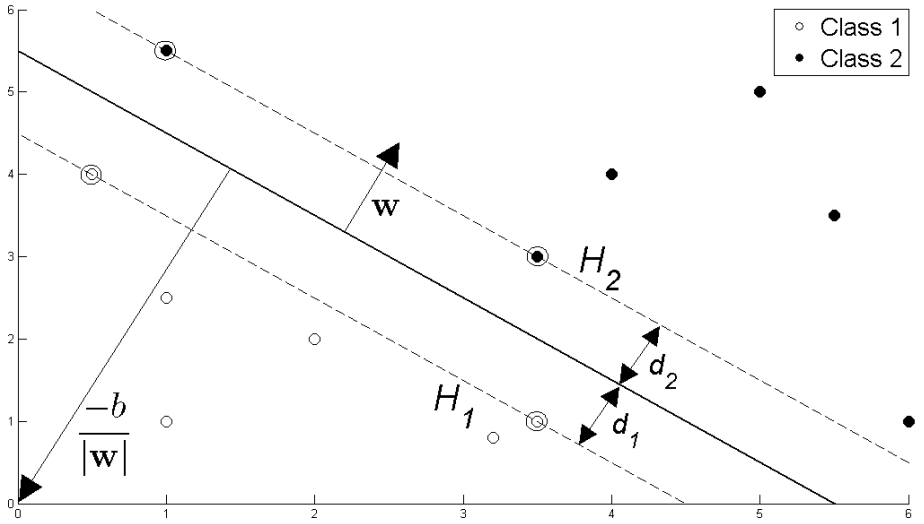
SVM is classsifier that can process high-dimension data, without a significant loss in performance. SVM is now more widely used. In information retrieval, SVM has also been widely used, especially in the classification process. SVM ability to process high-dimension data is very suitable to be applied to the text data, which is has high-dimension data. In this paper discussed the SVM performance comparisons with other classsifier performance in information retrieval. The discussion is based on the results of studies of several researchers. In one study it was found that SVM outperformed other classsifier. In the other study found SVM performance comparable with other classsifier. At the end of the paper, also discussed about the steps of text classification using SVM. This text classification process used in information retrieval of text data.
Keywords : SVM, *Information retrieval*, *classsifier.*

1. PEDAHULUAN

SVM (Support Vector Machine) adalah salah satu *classsifier* yang sekarang banyak digunakan untuk melakukan berbagai keperluan klasifikasi. Selain untuk klasifikasi, SVM juga digunakan untuk regresi. SVM merupakan *binary classifier* yang membagi data menjadi dua *class* dengan sebuah *hyperplane*. *Hyperplane* ini tepat berada di tengah-tengah kedua *class* dengan jarak *d* ke titik data terdekat untuk masing-masing *class*. *d* disebut *margin*, dan titik-titik data yang berada tepat pada jarak *d* dari *hyperplane* disebut *support vector*. Hyperplane SVM dinyatakan dengan persamaan sebagai berikut.

$$w \bullet x + b = 0$$

dimana w merupakan *normal* dari *hyperplane*, dan $\frac{b}{\|w\|}$ adalah jarak *hyperplane* ke titik *origin*. Gambar 1 memperlihatkan *hyperplane* yang membagi dua buah *class*.



Gambar 1: Hyperplane membagi dua buah class. (sumber gambar: **Fletcher (2009)**)

Titik-titik data yang masuk ke *class* 1 adalah titik-titik data yang memenuhi persamaan

$$w \bullet x + b \leq -1$$

dan titik-titik data yang masuk ke class 2 adalah titik-titik data yang memenuhi persamaan

$$w \bullet x + b \geq 1$$

Pada information retrieval SVM juga banyak digunakan, terutama pada proses klasifikasi data. Kemampuannya untuk mengolah data berdimensi besar menjadi keunggulan SVM dibanding dengan *classsifier* lain. Pada *information retrieval* data teks, keunggulan SVM untuk mengolah data berdimensi besar dapat dimanfaatkan, karena sifat data teks yang biasanya berdimensi besar. Pada tulisan ini akan dibahas perbandingan performa SVM dengan performa *classsifier* lain pada *information retrieval*. Perbandingan dibahas berdasarkan hasil penelitian beberapa peneliti. Di akhir tulisan juga dibahas langkah-langkah klasifikasi data teks. Proses klasifikasi ini digunakan pada proses *information retrieval* data teks.

2. PEMBAHASAN

2.1. SVM pada Information Retrieval dan Perbandingannya dengan algoritma-algoritma lain

Pada saat sekarang SVM telah banyak diterakan pada bidang *information retrieval*. Sebagian besar penggunaan SVM pada *information retrieval* adalah pada pengklasifikasian dan pengkategorian dokumen, serta pada proses *searching* yang menerapkan *relevancy feedback*. Selain penggunaan pada dua hal tersebut, SVM juga digunakan untuk proses perengkingan pada *document retrieval*. Kemampuan SVM yang cepat walaupun untuk data berdimensi tinggi kadang-kadang menjadi solusi yang tepat untuk keperluan *information retrival* yang membutuhkan kecepatan.

Joachims (1998) mengenalkan SVM untuk *text categorization*. Di dalam tulisannya Joachims memberikan bukti secara teoritis dan secara eksperimen bahwa SVM sangat cocok untuk *text categorization*. Secara teoritis Joachims mengemukakan beberapa alasan mengapa SVM cocok digunakan untuk *text categorization*. Beberapa alasannya adalah sebagai berikut.

1. *High dimensional input space*: Pada *text categorization* akan didapati jumlah fitur yang sangat besar (lebih dari 10000), dan SVM cenderung tidak tergantung pada besarnya dimensi data.
2. *Few irrelevant features*: Karena sangat sedikit fitur-fitur yang tidak relevan, maka pemilihan fitur untuk tujuan mereduksi dimensi menjadi tidak efektif.
3. *Document vectors are sparse*: Vector-vector yang mewakili dokumen hanya memiliki sedikit bagian yang tidak bernilai 0. Kivinen, Warmuth, dan Auer (1995) memberikan bukti secara teoritis dan empiris, bahwa algoritma seperti SVM sangat cocok untuk menyelesaikan permasalahan seperti ini.
4. *Most text categorization problems are linearly separable*: Semua kategori dari data *Ohsumed* terpisah secara linear, begitu juga sebagian besar data *Reuters*. Ide dasar SVM adalah untuk medapatkan pemisah linear seperti itu.

Secara eksperimen, Joachims membandingakn SVM dengan beberapa algoritma lain yaitu, *Bayes*, *Rocchio*, *R4.5*, dan *k-NN*. Dari hasil eksperimen diketahui bahwa SVM menghasilkan performa yang baik, mengungguli algoritma-algoritma lainnya secara substansial dan signifikan.

Pada *information retrieval*, SVM juga digunakan pada proses *relevance feedback*. Pada *information retrieval* yang menggunakan *relevancy feedback*, *feedback* yang diberikan oleh user digunakan sebagai data training SVM, dimana selanjutnya *classifier* yang dihasilkan digunakan untuk menghasilkan *result* baru dengan tingkat relefansi lebih

tinggi. Proses pemberian *feedback* ini bisa dilakukan secara iteratif. Drucker, Shahrary, dan Gibbon (2002) membandingkan SVM dengan beberapa algoritma lain untuk *information retrieval* menggunakan *relevancy feedback*.

Drucker, Shahrary, dan Gibbon (2002) membandingkan SVM dengan *Rocchio*, *Ide*, dan *Ide dec-hi* pada *information retrieval* dokumen teks menggunakan *relevancy feedback*. Dari kesimpulan yang diambil, Drucker, Shahrary, dan Gibbon kemudian memberikan 3 buah rekomendasi sebagai berikut.

1. Jika menggunakan *TF-IDF weighting*, dan dapat dipastikan bahwa pencarian awal dapat menghasilkan banyak dokumen yang relevan, maka SVM secara tipis lebih baik dari *Ide dec-hi*. Namun bagaimanapun, algoritma *Ide dec-hi* lebih simpel, lebih cepat, dan tidak perlu khawatir terhadap konvergensi algoritma SVM. Tidak pernah ada masalah dengan konvergensi, tapi hal itu bisa saja terjadi.
2. Jika lebih memilih untuk tidak menggunakan fitur *TF-IDF*, gunakan SVM dengan *binary fiature weighting*.
3. Gunakan SVM jika tidak bisa dipastikan seberapa berhasil pencarian awal yang akan dilakukan, karena jika pencarian awal jelek, maka algoritma-algoritma selain SVM akan memunyai performa yang buruk.

Pada *information retrieval*, SVM juga digunakan untuk proses perengkingan dokumen. Perengkingan dokumen berbasis SVM dikenalkan oleh Herbrich, Graepel, dan Obermayer (1999). Cao dkk.(2006) yang mengacu metode yang dikenalkan oleh Herbrich, Graepel, dan Obermayer sebagai *Ranking SVM* (RSVM) menggunakannya kembali untuk *information retrieval* dengan menambahkan dua pilihan optimasi, yaitu *gradient descent* dan *quadratic programming*, dan menamai metodenya *Ranking SVM for IR* (RSVM-IR).

Cao dkk. membandingkan *RSVM-IR* dengan *BM25*, dan *language model for information retrieval* (LMIR). Dari hasil percobaan yang dilakukan diambil kesimpulan bahwa *RSVM-IR* mengungguli *Ranking SVM* dan algoritma lainya secara signifikan.

Colas dan Brazdil (2006) membandingkan SVM dengan *k-NN* dan *Naive Bayes*. Algoritma-algoritma dibandingkan dalam versi optimasi masing-masing. Hasil yang didapatkan menunjukkan bahwa semua algoritma mendapatkan performa yang sebanding di sebagian besar permasalahan. Satu hasil yang mengejutkan adalah SVM bukan merupakan pemenang sejati, meskipun cukup baik untuk performa keseluruhan. Jika *preprocessing* yang sesuai diterapkan pada *k-NN*, maka algoritma ini mendapatkan hasil

yang sangat bagus secara terus menerus. *Naive Bayes* juga mendapatkan performa yang bagus.

2.2. Langkah-langkah klasifikasi teks menggunakan SVM

Proses klasifikasi teks dengan menggunakan SVM hampir sama dengan klasifikasi objek lain dengan menggunakan SVM. Perbedaannya terletak pada proses ekstraksi fitur. Pada *information retrieval*, proses ekstraksi fitur dikenal sebagai proses *weighting*. Secara garis besar langkah-langkah klasifikasi teks dengan SVM adalah *indexing*, *weighting*, *training*, *dan testing.*

2.2.1 Indexing

Indexing merupakan suatu *preprocessing* dalam klasifikasi teks. Jika dokumen yang akan diklasifikasi sudah mengalami *indexing* sebelumnya, maka proses klasifikasi cukup menggunakan index dokumen untuk ekstraksi fitur. Jika dokumen belum di index, maka pada proses klasifikasi diperlukan proses *searching* dan pengorganisasian *terms* yang prosesnya tidak sederhana. Proses *indexing* penting karena *weighting* menggunakan ada tidaknya atau jumlah suatu *term* pada dokumen.

2.2.2 Weighting

Proses *weighting* adalah kata lain dari ekstraksi fitur. Ada beberapa macam cara *weighting*, yaitu, *binary weighting*, *term frequency* (TF), *inverse document frequency* (IDF), dan perpaduan antara *TF* dan *IDF* disebut *TF-IDF*.

Binary weighting adalah pembentukan vektor fitur dengan melihat ada tidaknya suatu *term* di dokumen tersebut. *TF* adalah pembentukan vektor fitur dari jumlah suatu *term* pada suatu dokumen. *IDF* adalah *TF* dikalikan dengan $Log(N/n_i)$ dimana N adalah jumlah keseluruhan dokumen, dan n_i adalah jumlah dokumen yang mengandung *term* tersebut. *TF-IDF* adalah kombinasi *TF* dan *IDF* dimana *TF* dikalikan oleh *IDF*.

Pertama-tama ruang vektor fitur dibentuk dengan menghitung jumlah jenis kata yang terdapat di seluruh dokumen. Jika pada keseluruhan dokumen terdapat 1000 yang berbeda, maka vektor fitur akan berdimensi 1000, dimana setiap bagian vektor mewakili satu kata. Dimensi vektor dapat direduksi dengan cara mem-*prunning* kata-kata yang tidak signifikan.

Setelah ruang vektor terbentuk, selanjutnya vektor-vektor yang mewakili suatu dokumen dibentuk dengan menggunakan salah satu metode *weighting*. Misalkan dari koleksi dokumen yang terdiri dari 500 dokumen ditemukan 1000 jenis kata yang berbeda,

jika tidak dilakukan *prunning*, maka akan didapatkan ruang vector berdimensi 1000. Misalkan dimensi pertama mewakili kata ”bola”, dan dimensi ke-2 mewakili kata ”buku”, maka jika pada dokumen pertama terdapat kata ”bola” dan tidak terdapat kata ”buku”, jika menggunakan *binary weighting*, maka vector yang mewakili dokumen pertama dimensi pertamanya akan bernilai 1 dan dimensi ke-2 nya bernilai 0. Jika dimensi pertama disebut x_1 dan dimensi ke-2 disebut x_2 maka $x_1 = 1$, dan $x_2 = 0$, dan seterusnya. Jika pada dokumen pertama terdapat 25 buah kata ”bola” dan 5 buah kata”buku”, jika menggunakan *TF* maka vektor fitur untuk dokumen pertama $x_1 = 25$, dan $x_2 = 5$. *IDF* jarang dipergunakan karena prosesnya yang lebih komplek. Pada *IDF* vektor fitur baru dapat dibentuk setelah melakukan pernghitungan pada seluruh dokumen.

2.2.3 Training

Setelah vektor-vektor fitur terbentuk, lengkap dengan labelnya masing-masing, maka vector-vector tersebut telah siap dimasukkan ke SVM untuk dijadikan data *training*. Yang perlu dijadikan perhatian pada proses *training* adalah permasalahan *tunning* terhadap parameter-parameter SVM. Penentuan parameter yang tepat akan menghasilkan *classifier* yang lebih baik. Namun permasalahannya adalah, sampai saat ini belum ada teori yang bisa menjadi dasar bagaimana cara untuk menentukan parameter-parameter SVM secara tepat. Selama ini cara yang dilakukan adalah metode coba-coba melalui proses iterasi. Selain itu untuk menentukan jenis *kernel* yang dipakaipun seharusnya terlebih dahulu harus diketahui sifat-sifat data, namun pada kenyataannya hal ini jarang terjadi. Sehingga untuk data yang tidak diketahui sifatnya, para peneliti lebih memilih menggunakan *RBF kernel*.

Hasil keluaran SVM dari proses *training* adalah nilai *alpha* untuk setiap vektor dan sebuah nilai *b*. Untuk vektor yang bukan merupakan *support vectors*, nilai *alpha*-nya akan bernilai 0. *Classifier* akan dibentuk oleh nilai-nilai *alpha* dan nilai *b*.

2.2.4 Testing

Pada saat *testing* diperlukan nilai-nilai *alpha* dan nilai *b* yang didapatkan pada saat *training* untuk melakukan proses klasifikasi. Vektor-vektor fitur yang dijadikan data *testing* dimasukkan ke SVM tanpa disertai labelnya beserta nilai-nilai *alpha* dan *b*. Keluaran hasil *testing* berupa label *class* hasil klasifikasi. Untuk mengecek tingkat akurasi *classifier*, label keluaran ini kemudian dibandingkan dengan label aslinya.

2.2.5 Masalah Multiclass

Sebagian besar permasalahan klasifikasi yang dijumpai pada klasifikasi teks bersifat *binary classification*, misalnya pada proses *filltering*, yang dibutuhkan adalah jawaban ya atau tidak, pada *relevancy feedback* yang diperlukan adalah jawaban relevan atau tidak. Namun dalam beberapa kasus, permasalahan *multiclass* bisa muncul. Melihat sifat dasar SVM yang merupakan sebuah *binary classifier*, maka diperlukan suatu sekenario untuk menggunakan SVM sebagai *multiclass classifier*.

Ada dua sekenario yang sering digunakan, yaitu sekenario *one against one*, dan *one against all*. Pada sekenario *one against one* dibuat sejumlah *classifier* yang memisahkan satu *class* dengan satu *class* lain. Setiap pasangan *class* diadu satu sama lain, *class* yang paling menang dijadikan sebagai *class* label akhir. Pada sekenario *one against all*, dibuat sejumlah *classifier* yang memisahkan satu *class* dengan semua *class* lainnya. Di saat data masuk ke suatu *class* dimana menghasilkan jarak yang paling jauh dari *hyperplane*, maka kelas tersebut dijadikan *class label* data tersebut. Untuk jumlah kelas yang banyak, sekenario *one against one* sulit untuk dilakukan, karena akan diperlukan banyak sekali *classifier*.

2.2.6 Masalah Perengkingan

Masalah perengkingan biasanya timbul pada saat pengembalian suatu hasil *query*. Kadang kala, tidak semua hasil *query* yang bisa didapatkan oleh mesin disuguhkan kepada *user*, dan mungkin hanya sebagian kecilnya saja. Dari itu diusahakan supaya hasil yang ditampilkan kepada *user* adalah hasil yang paling baik. Untuk itu diperlukan suatu proses perengkingan.

Proses perengkingan pada pengembalian hasil *query* mirip dengan proses sekenario *one against all* pada kasus klasifikasi *multiclas*s. Bedanya adalah pada proses perengkingan ini hanya ada satu *classifier*. Nilai rengking ditentukan dari seberapa jauh letak vektor fitur dari *hyperplane*. Semakin jauh letak vektor fitur dari *hyperplane*, semakin jelas bahwa data tersebut masuk ke *class* tersebut, maka semakin baik rengking nya.

3. PENUTUP

SVM dapat diterapkan pada berbagai bagian dalam *information retrieval*, seperti pengklasifikasian dan pengkategorian dokument atau teks, proses *relefancy feedback*, dan perengkingan dokumen. Kemampuan SVM yang tidak terpengaruh dimensi data sangat cocok untuk menyelesaikan masalah-masalah yang berkaitan dengan dimensi besar pada

information retrieval yang tidak bisa ditanggulangi oleh algoritma-algoritma lain. Dari hasil beberapa percobaan yang dilakukan oleh beberapa peneliti, dalam beberapa kasus, SVM sangat singnifikan mengungguli algoritma lain. Dalam percobaan yang lain ditunjukkan bahwa SVM mempunyai performa berimbang dengan beberapa algoritma lain dalam beberapa kasus.

DAFTAR PUSTAKA

Cao, Y., dkk. 2006. *Adapting Ranking SVM to document retrieval*. SIGIR '06 Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. Page 186 – 193.

Colas, F., Brazdil, P. 2006. *Comparison of SVM and some older classification algorithms in text classification tasks*. Artificial Intelligence in Theory and Practice. Page 169-178. Springer US.

Drucker, H., Shahrary, B., Gibbon, D. C. 2002. *Support Vector Machines: Relevance feedback and information retrieval*. Information Processing and Management 38, 305-323. Pergamon.

Fletcher, T. 2009. *Support Vector Machine explained*. UCL. www.cs.ucl.ac.uk/staff/T. Fletcher/.

Herbrich, R., Graepel, T., Obermayer, K. 1999. *Large margin rank boundaries for ordinal regression*. Advances in neural information processing systems. Page 115 – 132. MIT.

Joachims, T. 1998. *Text categorization with Support Vector Machines: Learning with many relevant features*. Machine Learning: ECML-98, Lecture Notes in Computer Science Volume 1398, 1998, pp 137-142. Springer Berlin Heidelberg.

Kivinen, J., Warmuth, M. K., Auer, P. 1995. *The Perceptron algorithm versus Winnow: linear versus logarithmic mistake bounds when few input variables are relevant*. Artificial Intelligence 97 (1997) 325-343. Elsevier.