A Simple Introduction to Support Vector Machines

Adapted from various authors
by Mario Martin

Outline

- Large-margin linear classifier
  - Linear separable
  - Nonlinear separable
- Creating nonlinear classifiers: kernel trick
- Transduction
- Discussion on SVM
- Conclusion

History of SVM

- SVM is related to statistical learning theory [3]
- Introduced by Vapnik
- SVM was first introduced in 1992
- SVM becomes popular because of its success a lot of classification problems

SVM: Large-margin linear classifier
Perceptron Revisited: Linear Separators

- Binary classification can be viewed as the task of separating classes in feature space:

\[ w^T x + b = 0 \]

\[ w^T x + b > 0 \]

\[ w^T x + b < 0 \]

\[ f(x) = \text{sign}(w^T x + b) \]

May 13, 2012

Simple introduction to SVMs

Linear Separators

- Which of the linear separators is optimal?

May 13, 2012

Simple introduction to SVMs

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
  - Other different algorithms have been proposed
- Are all decision boundaries equally good?

Examples of Bad Decision Boundaries
Maximum Margin Classification

- Maximizing the distance to examples is good according to intuition and PAC theory.
- Implies that only few vectors matter; other training examples are ignorable.

Classification Margin

- Distance from example $x_i$ to the separator is $r = \frac{w^T x_i + b}{||w||}$
- Examples closest to the hyperplane are support vectors.
- Margin $\rho$ of the separator is the distance between support vectors.

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, $m$

We normalize equations so function in supports is 1/-1.

Finding the Decision Boundary

- Let $\{x_1, ..., x_n\}$ 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 $\Rightarrow y_i(w^T x_i + b) \geq 1$, $\forall i$

- Maximizing margin classifying all points correctly constraints is defined as follows:
Finding the Decision Boundary

- Primal formulation
  
  Minimize $\frac{1}{2}\|w\|^2$
  subject to $y_i(w^T x_i + b) \geq 1 \quad \forall i$

- We can solve this problem using this formulation, or using the dual formulation...

[Recap of Constrained Optimization]

- Suppose we want to: minimize $f(x)$ subject to $g(x) = 0$
- A necessary condition for $x_0$ to be a solution:
  
  \[
  \frac{\partial}{\partial x} \left( f(x) + \alpha g(x) \right) \bigg|_{x=x_0} = 0 \\
  g(x) = 0 \\
  \alpha: \text{the Lagrange multiplier}
  \]
- For multiple constraints $g_i(x) = 0$, $i=1, \ldots, m$, we need a Lagrange multiplier $\alpha_i$ for each of the constraints
  
  \[
  \frac{\partial}{\partial x} \left( f(x) + \sum_{i=1}^m \alpha_i g_i(x) \right) \bigg|_{x=x_0} = 0 \\
  g_i(x) = 0 \quad \text{for } i = 1, \ldots, m
  \]

Back to the Original Problem

Minimize $\frac{1}{2}\|w\|^2$
subject to $1 - y_i(w^T x_i + b) \leq 0$ for $i = 1, \ldots, n$

- The Lagrangian is
  
  $L = \frac{1}{2}w^Tw + \sum_{i=1}^n \alpha_i \left( 1 - y_i(w^T x_i + b) \right)$

- Note that $\|w\|^2 = w^Tw$
- Setting the gradient of $L$ w.r.t. $w$ and $b$ to zero, we have
  
  $w + \sum_{i=1}^n \alpha_i (-y_i)x_i = 0 \quad \Rightarrow \quad w = \sum_{i=1}^n \alpha_i y_i x_i$
The Dual Formulation

- If we substitute \( w = \sum_{i=1}^{n} \alpha_i y_i x_i \) to \( \mathcal{L} \), we have

\[
\mathcal{L} = \frac{1}{2} \sum_{i=1}^{n} \alpha_i y_i x_i^T \sum_{j=1}^{n} \alpha_j y_j x_j + \sum_{i=1}^{n} \alpha_i \left( 1 - y_i \left( \sum_{j=1}^{n} \alpha_j y_j x_j^T x_i + b \right) \right)
\]

\[
= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n} \alpha_i \sum_{j=1}^{n} \alpha_j y_j x_j^T x_i \quad b \sum_{i=1}^{n} \alpha_i y_i
\]

- Remember that \( \sum_{i=1}^{n} \alpha_i y_i = 0 \)

- This is a function of \( \alpha_i \) only

The Dual formulation

- It is known as the dual problem (the original problem is known as the primal problem): if we know \( w \), we know all \( \alpha_i \); if we know all \( \alpha_i \), we know \( w \)

- The objective function of the dual problem needs to be maximized!

- The dual problem is therefore:

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

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

Properties of \( \alpha_i \) when we introduce the Lagrange multipliers

The result when we differentiate the original Lagrangian w.r.t. \( b \)

The Dual Problem

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

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

- This is a quadratic programming (QP) problem

- A global maximum of \( \alpha_i \) can always be found

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

A Geometrical Interpretation

\[
\begin{align*}
\alpha_1 &= 0.8 \\
\alpha_2 &= 0 \\
\alpha_3 &= 0 \\
\alpha_4 &= 0 \\
\alpha_5 &= 0 \\
\alpha_6 &= 0.6 \\
\alpha_7 &= 0 \\
\alpha_8 &= 1.4 \\
\alpha_9 &= 0 \\
\alpha_{10} &= 0
\end{align*}
\]

\( w^T x + b = 1 \)

\( w^T x + b = -1 \)
Characteristics of the Solution

- Many of the $\alpha_i$ are zero
- $w$ is a linear combination of a small number of data points
- This “sparse” representation can be viewed as data compression
- $x_i$ 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
  \[
  w = \sum_{j=1}^{s} \alpha_j y_t x_t
  \]

For testing with a new data $z$

- Compute $w^T z + b = \sum_{j=1}^{s} \alpha_j y_t (x_t^T z) + b$
- classify $z$ as class 1 if the sum is positive, and class 2 otherwise
- Note: $w$ need not be formed explicitly

The Quadratic Programming Problem

- Many approaches have been proposed
  - Loqo, cplex, etc. (see http://www.numerical.rl.ac.uk qp qp.html)
  - Most are “interior-point” methods
    - Start with an initial solution that can violate the constraints
    - Improve this solution by optimizing the objective function and/or reducing the amount of constraint violation
  - For SVM, sequential minimal optimization (SMO) seems to be the most popular
  - A QP with two variables is trivial to solve
  - Each iteration of SMO picks a pair of $(\alpha_i, \alpha_j)$ and solve the QP with these two variables; repeat until convergence
  - In practice, we can just regard the QP solver as a “black-box” without bothering how it works

Non-Separable Sets

- Sometimes, we do not want to separate perfectly.
- This point is too close!
- Maybe this point is not so important.
Non-Separable Sets

- Sometimes, we **do not** want to separate perfectly.

If we ignore this point

The hyperplane is nicer!

Soft Margin Classification

- What if the training set is not linearly separable?
- Slack variables $\xi_i$ can be added to allow misclassification of difficult or noisy examples, resulting margin called soft.

Non-linearly Separable Problems

- We allow “error” $\xi_i$ in classification; it is based on the output of the discriminant function $w^T x + b$
- $\xi_i$ approximates the number of misclassified samples

Class 1

Class 2

$w^T x + b = 1$

$w^T x + b = -1$
Soft Margin Hyperplane

- If we minimize $\sum_i \xi_i$, $\xi_i$ can be computed 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.
- Note that $\xi_i = 0$ if there is no error for $x_i$.
- Number of slacks + supports is an upper bound of the number of errors (Leave one out error).

We want to minimize

$$
\frac{1}{2} ||w||^2 + C \sum_{i=1}^n \xi_i
$$

- $C$: 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 $y_i(w^T x_i + b) \geq 1 - \xi_i$, $\xi_i \geq 0$

The Optimization Problem

- The dual of this new constrained optimization problem is

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

- $w$ is recovered as: $w = \sum_{j=1}^s \alpha_j y_j x_j$
- 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 $\alpha_i$

Non-linearly Separable Problems

- We allow “error” $\xi_i$ in classification; it is based on the output of the discriminant function $w^T x + b$.
- $\xi_i$ approximates the number of misclassified samples.

- $\xi_i$ approximates the number of misclassified samples.
SVM with KERNELS: Large-margin NON-linear classifiers

Extension to Non-linear Decision Boundary

- So far, we have only considered large-margin classifier with a linear decision boundary
- How to generalize it to become non-linear?
- Key idea: transform $x_i$ to a higher dimensional space to “make life easier”
  - Input space: the space the point $x_i$ are located
  - Feature space: the space of $\phi(x_i)$ 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 $x_1x_2$ makes 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

Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable

Note: feature space is of higher dimension than the input space in practice
The Kernel Trick

- Recall the SVM optimization problem

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

- 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

\[
K(x_i, x_j) = \phi(x_i)^T \phi(x_j)
\]

SVMs with kernels

- Training

\[
\begin{align*}
\text{maximize } &\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \cdot K(x_i, x_j) \\
\text{subject to } &\sum_{i=1}^{l} \alpha_i \cdot y_i = 0 \quad \text{and} \quad \forall i \ C \geq \alpha_i \geq 0
\end{align*}
\]

- Classification of \( x \):

\[
h(x) = \text{sign} \left( \sum_{i=1}^{l} \alpha_i \cdot y_i \cdot K(x_i, x) + b \right)
\]

An Example for \( \phi(.) \) and \( K(\ldots) \)

- Suppose \( \phi(.) \) is given as follows

\[
\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)
\]

- An inner product in the feature space is

\[
\langle \phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}), \phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) \rangle = (1 + x_1y_1 + x_2y_2)^2
\]

- 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

- Kernel (Gram) matrix:

\[
\begin{bmatrix}
K(x_1, x_1) & K(x_1, x_2) & K(x_1, x_3) & \cdots & K(x_1, x_l) \\
K(x_2, x_1) & K(x_2, x_2) & K(x_2, x_3) & & K(x_2, x_l) \\
& \cdots & & \ddots & \cdots \\
K(x_l, x_1) & K(x_l, x_2) & K(x_l, x_3) & & K(x_l, x_l)
\end{bmatrix}
\]

Matrix obtained from product:

\[
K = \phi' \phi
\]
Kernel Functions

- Any function $K(x, z)$ that creates a symmetric, positive definite matrix $K_{ij} = K(x_i, x_j)$ is a valid kernel (an inner product in some space).

- Why? Because any sdp matrix $M$ can be decomposed as $N'N = M$ so $N$ can be seen as the projection to the feature space.

Examples of Kernel Functions

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

- 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.
  - The feature space is infinite-dimensional.

- Sigmoid with parameter $\kappa$ and $\theta$
  \[ K(x, y) = \tanh(\kappa x^Ty + \theta) \]
  - 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,
  \[ \begin{align*}
  \text{Original:} & \quad \max_{\alpha_i} W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \\
  \text{subject to} & \quad C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0
  \end{align*} \]
  \[ \begin{align*}
  \text{With kernel function:} & \quad \max_{\alpha_i} W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\
  \text{subject to} & \quad C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0
  \end{align*} \]
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

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

More on Kernel Functions

- Since the training of SVM only requires the value of $K(x_i, x_j)$, there is no restriction of the form of $x_i$ and $x_j$
  - $x_i$ can be a sequence or a tree, instead of a feature vector
  - $K(x_i, x_j)$ is just a similarity measure comparing $x_i$ and $x_j$
  - For a test object $z$, the discriminant function essentially is a weighted sum of the similarity between $z$ and a pre-selected set of objects (the support vectors)
    $$ f(z) = \sum_{x_i \in S} \alpha_i y_i K(z, x_i) + b $$
    $S$: the set of support vectors

Choosing the Kernel Function

- Probably the most tricky part of using SVM.
- The kernel function is important because it creates the kernel matrix, which summarizes all the data
- Many principles have been proposed (diffusion kernel, Fisher kernel, string kernel, …)
- There is 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
- Note that SVM with RBF kernel is closely related to RBF neural networks, with the centers of the radial basis functions automatically chosen for SVM
Other Aspects of SVM

- How to use SVM for multi-class classification?
  - One can change the QP formulation to become multi-class
  - More often, multiple binary classifiers are combined
  - One can train multiple one-versus-all classifiers, or combine multiple pairwise classifiers “intelligently”
- How to interpret the SVM discriminant function value as probability?
  - By performing logistic regression on the SVM output of a set of data (validation set) that is not used for training
  - Some SVM software (like libsvm) have these features built-in

Software

- A list of SVM implementation can be found at http://www.kernel-machines.org/software.html
- Some implementation (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementation of SVM
- Several Matlab toolboxes for SVM are also available

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

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

**Weaknesses**

- Need to choose a “good” kernel function.
Other Types of Kernel Methods

- A lesson learnt in SVM: a linear algorithm in the feature space is equivalent to a non-linear algorithm in the input space
- Standard linear algorithms can be generalized to its non-linear version by going to the feature space
- Kernel principal component analysis, kernel independent component analysis, kernel canonical correlation analysis, kernel k-means, 1-class SVM are some examples

Conclusion

- SVM is a useful alternative to neural networks
- Two key concepts of SVM: maximize the margin and the kernel trick
- Many SVM implementations are available on the web for you to try on your data set!

Toy Examples

- All examples have been run with the 2D graphic interface of SVMLIB (Chang and Lin, National University of Taiwan)

"LIBSVM is an integrated software for support vector classification, (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM). It supports multi-class classification. The basic algorithm is a simplification of both SMO by Platt and SVMLight by Joachims. It is also a simplification of the modification 2 of SMO by Keerthy et al. Our goal is to help users from other fields to easily use SVM as a tool. LIBSVM provides a simple interface where users can easily link it with their own programs..."

- Available from: www.csie.ntu.edu.tw/~cjlin/libsvm (it includes a Web integrated demo tool)
Toy Examples (I)

(still) Linearly separable data set
Linear SVM
High value of C parameter
Maximal margin Hyperplane

The example is correctly classified

Toy Examples (I)

(still) Linearly separable data set
Linear SVM
Low value of C parameter
Trade-off between: margin and training error

The example is now a bounded SV

Toy Examples (I)

Toy Examples (I)
Examples

Toy Examples (I)

Resources

- http://www.kernel-machines.org/
- http://www.support-vector.net/
The learning problem

- **Transduction:**
  - We consider a phenomenon \( f \) that maps inputs (instances) \( x \) to outputs (labels) \( y = f(x) \ (y \in \{-1, 1\}) \)
    - Given a set of labeled examples \( \{(x_i, y_i) : i = 1, \ldots, n\} \),
    - and a set of unlabeled examples \( x'_{1}, \ldots, x'm \)
    - the goal is to find the labels \( y'_{1}, \ldots, y'm \)
  - No need to construct a function \( f \), the output of the transduction algorithm is a vector of labels.

Transduction based on margin size

- Binary classification, linear parameterization, joint set of (training + working) samples

- **Two objectives of transductive learning:**
  - (TL1) separate labeled training data using a large-margin hyperplane (as in standard inductive SVM)
  - (TL2) separating (explain) working data set using a large-margin hyperplane.

Transductive SVMs

- **Transductive** instead of inductive (Vapnik 98)
- TSVMs take into account a particular test set and try to minimize misclassifications of just those particular examples
- Formal setting:
  \[
  S_{\text{train}} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}
  \]
  \[
  S_{\text{test}} = \{x'_{1}, \ldots, x'_{k}\} \text{ (normally } k \gg n)\]
- Goal of the transductive learner \( L \):
  find a function \( h_L = L(S_{\text{train}}, S_{\text{test}}) \) so that the expected number of erroneous predictions on the test examples is minimized.
Optimization formulation for SVM transduction

- **Given:** joint set of (training + working) samples
- **Denote slack variables** for training, for working
- **Minimize**
  \[ R(w, b) = \frac{1}{2}(w \cdot w) + C \sum_{i=1}^{n} \xi_i + C^* \sum_{j=1}^{m} \xi_j^* \]
- **subject to**
  \[ y_i [ (w \cdot x_i) + b ] \geq 1 - \xi_i \]
  \[ y_j^* [ (w \cdot x_j) + b ] \geq 1 - \xi_j^* \]
  \[ \xi_i, \xi_j^* \geq 0, i = 1, ..., n, j = 1, ..., m \]

- **Solution (~ decision boundary)**
  \[ D(x) = (w^* \cdot x) + b^* \]
- **Unbalanced situation (small training/ large test)**
  - all unlabeled samples assigned to one class
- **Additional constraint:**
  \[ \frac{1}{n} \sum_{i=1}^{n} y_i = \frac{1}{m} \sum_{j=1}^{m} [ (w \cdot x_j) + b ] \]

Optimization formulation (cont’d)

- **Hyperparameters** \( C \) and \( C^* \) control the trade-off between explanation and margin size
- **Soft-margin inductive SVM** is a special case of soft-margin transduction with zero slacks \( \xi_j^* = 0 \)
- **Dual + kernel** version of SVM transduction
- **Transductive SVM optimization** is **not convex**
  (~ non-convexity of the loss for unlabeled data) –
  - different opt. heuristics ~ different solutions
- **Exact solution** (via exhaustive search) possible for **small number** of test samples \( m \)
Many applications for transduction

- **Text categorization**: classify word documents into a number of predetermined categories
- **Email classification**: Spam vs non-spam
- **Web page classification**
- **Image database classification**
- All these applications:
  - high-dimensional data
  - small labeled training set (human-labeled)
  - large unlabeled test set

Example application

- Prediction of molecular bioactivity for drug discovery
- Training data \(\approx 1,909\); test \(\approx 634\) samples
- Input space \(\approx 139,351\)-dimensional
- Prediction accuracy:
  - SVM induction \(\approx 74.5\%\)
  - transduction \(\approx 82.3\%\)

Ref: J. Weston et al, KDD cup 2001 data analysis: prediction of molecular bioactivity for drug design—binding to thrombin, *Bioinformatics 2003*