Lecture Slides for

INTRODUCTION TO

Machine Learning

ETHEM ALPAYDIN
© The MIT Press, 2004

alpaydin@boun.edu.tr
http://www.cmpe.boun.edu.tr/~ethem/i2ml
Likelihood- vs. Discriminant-based Classification

- **Likelihood-based:** Assume a model for $p(x|C_i)$, use Bayes’ rule to calculate $P(C_i|x)$
  
  $Choose C_i if \quad g_i(x) = \log P(C_i|x) \ is \ maximum$

- **Discriminant-based:** Assume a model for the discriminant $g_i(x|\Phi_i)$; no density estimation
  
  □ Estimating the boundaries is enough; no need to accurately estimate the densities inside the boundaries
Linear Discriminant

- Linear discriminant:

\[ g_i(x \mid w_i, w_{i0}) = w_i^T x + w_{i0} = \sum_{j=1}^{d} w_{ij}x_j + w_{i0} \]

- Advantages:
  - Simple: \(O(d)\) space/computation
  - Knowledge extraction: Weighted sum of attributes; positive/negative weights, magnitudes (credit scoring)
  - Optimal when \(p(x|C_i)\) are Gaussian with shared cov matrix; useful when classes are (almost) linearly separable
Generalized Linear Model

- **Quadratic discriminant:**
  \[ g_i(x \mid W_i, w_i, w_{i0}) = x^T W_i x + w_i^T x + w_{i0} \]

- Instead of higher complexity, we can still use a linear classifier if we use higher-order (product) terms.

- Map from \( x \) to \( z \) using **nonlinear basis functions** and use a linear discriminant in \( z \)-space
  \[ Z_1 = x_1, \ Z_2 = x_2, \ Z_3 = x_1^2, \ Z_4 = x_2^2, \ Z_5 = x_1 x_2 \]

- The linear function defined in the \( z \) space corresponds to a **non-linear** function in the \( x \) space.
  \[ g_i(x) = \sum_{j=1}^{k} w_{ij} \phi_j(x) \]
**Two Classes**

Choose $C_1$ if $g_1(x) > g_2(x)$

Choose $C_2$ if $g_2(x) > g_1(x)$

Define:

$$g(x) = g_1(x) - g_2(x)$$

$$= (w_1^T x + w_{10}) - (w_2^T x + w_{20})$$

$$= (w_1 - w_2)^T x + (w_{10} - w_{20})$$

$$= w^T x + w_0$$

Choose

$$\begin{cases}
C_1 & \text{if } g(x) > 0 \\
C_2 & \text{otherwise}
\end{cases}$$
Learning the Discriminants

As we have seen before, when $p(x \mid C_i) \sim \mathcal{N}(\mu_i, \Sigma)$, the optimal discriminant is a linear one:

$$g_i(x \mid w_i, w_{i0}) = w_i^T x + w_{i0}$$

$$w_i = \Sigma^{-1}\mu_i \quad w_{i0} = -\frac{1}{2}\mu_i^T\Sigma^{-1}\mu_i + \log P(C_i)$$

So, estimate $\mu_i$ and $\Sigma$ from data, and plug into the $g_i$’s to find the linear discriminant functions.

Of course any way of learning can be used (e.g. perceptron, gradient descent, logistic regression...).
When $K > 2$

- Combine $K$ two-class problems, each one separating one class from all other classes
Multiple Classes

\[ g_i(x \mid w_i, w_{i0}) = w_i^T x + w_{i0} \]

How to train?
How to decide on a test?

Choose \( C_i \) if
\[ g_i(x) = \max_{j=1}^{K} g_j(x) \]

Why? Any problem?

Convex decision regions based on \( g_i \)s (indicated with blue);
dist is \( |g_i(x)|/||w_i|| \)

Assumes that classes are linearly separable:
reject may be used
Pairwise Separation

If the classes are not linearly separable:

\[ g_{ij}(x | w_{ij}, w_{ij0}) = w_{ij}^T x + w_{ij0} \]

\[
\begin{cases}
  > 0 & \text{if } x \in C_i \\
  \leq 0 & \text{if } x \in C_j \\
  \text{don't care} & \text{otherwise}
\end{cases}
\]

choose \( C_i \) if \( \forall j \neq i, g_{ij}(x) > 0 \)

Uses \( k(k-1)/2 \) linear discriminants
- Pairwise linear separation is much more likely than linear separability.
- None of the classes may satisfy the condition
  - Reject
  - Use max

\[
\text{choose } C_i \text{ if } \forall j \neq i, \ g_{ij}(x) > 0 \quad \rightarrow \quad \text{choose } C_i \text{ maximizing } \ g_i(x) = \sum_{j \neq i} g_{ij}(x)
\]
Dot Product and Projection

\[ \langle w, p \rangle = w^T p = ||w|| ||p|| \cos \theta \]

proj. of \( p \) onto \( w \)

\[ = ||p|| \cos \theta \]

\[ = w^T . p / ||w|| \]
Geometry

The points $x$ on the separating hyperplane have $g(x) = \mathbf{w}^T \mathbf{x} + w_0 = 0$. Hence for the points on the boundary $\mathbf{w}^T \mathbf{x} = -w_0$.

Thus, these points also have the same projection onto the weight vector $\mathbf{w}$, namely $\mathbf{w}^T \mathbf{x} / ||\mathbf{w}||$ (by definition of projection and dot product). But this is equal to $-w_0 / ||\mathbf{w}||$. Hence ...

The perpendicular distance of the boundary to the origin is $|w_0|/||\mathbf{w}||$.

The distance of any point $x$ to the decision boundary is $|g(x)|/||\mathbf{w}||$. 
Support Vector Machines
Vapnik and Chervonenkis – 1963
Boser, Guyon and Vapnik – 1992 (kernel trick)
Cortes and Vapnik – 1995 (soft margin)

The SVM is a machine learning algorithm which
- solves classification problems
- uses a flexible representation of the class boundaries
- implements automatic complexity control to reduce overfitting
- has a single global minimum which can be found in polynomial time

It is popular because
- it can be easy to use
- it often has good generalization performance
- the same algorithm solves a variety of problems with little tuning
SVM Concepts

- Convex programming and duality
- Using maximum margin to control complexity
- Representing non-linear boundaries with feature expansion
- The kernel trick for efficient optimization
Which of the linear separators is optimal?
Classification Margin

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

![Diagram showing classification margin and support vectors]
Maximizing the margin is good according to intuition.

Implies that only support vectors matter; other training examples are ignorable.
SVM as 2-class Linear Classifier

(Cortes and Vapnik, 1995; Vapnik, 1995)

\[ X = \{ x^t, r^t \}, \text{ where } r^t = \begin{cases} +1 & \text{if } x^t \in C_1 \\ -1 & \text{if } x^t \in C_2 \end{cases} \]

find \( w \) and \( w_0 \) such that

\[ w^T x^t + w_0 \geq 1 \text{ for } r^t = +1 \]

\[ w^T x^t + w_0 \leq -1 \text{ for } r^t = -1 \]

Note the condition \( \geq 1 \) (not just \( \geq 0 \)). We can always do this if the classes are linearly separable by rescaling \( w \) and \( w_0 \), without affecting the separating hyperplane:

\[ w^T x^t + w_0 = 0 \]

Optimal separating hyperplane: Separating hyperplane maximizing the margin
Optimal Separating Hyperplane

Must satisfy:
\[ w^T x^t + w_0 \geq +1 \quad \text{for} \quad r^t = +1 \]
\[ w^T x^t + w_0 \leq -1 \quad \text{for} \quad r^t = -1 \]
which can be rewritten as
\[ r^t (w^T x^t + w_0) \geq +1 \]

(Cortes and Vapnik, 1995; Vapnik, 1995)
Maximizing the Margin

Distance from the discriminant to the closest instances on either side is called the margin.

In general this relationship holds (geometry):

$$d = \frac{|g(x)|}{\|w\|}$$

So, for the support vectors, we have:

$$d = \begin{cases} \frac{1}{\|w\|} \\ \frac{|-1|}{\|w\|} \end{cases}$$

$$\rho = 2d = \frac{2}{\|w\|}$$

To maximize margin, minimize the Euclidian norm of the weight vector \(w\).
Maximizing the Margin - Alternate explanation

- Distance from the discriminant to the closest instances on either side is called the margin.

- Distance of $x$ to the hyperplane is
  \[ \frac{|w^T x + w_0|}{\|w\|} \]

- We require that this distance is at least some value $\rho > 0$.

- We would like to maximize $\rho$, but we can do so in infinitely many ways by scaling $w$.

- For a unique solution, we fix $\rho \|w\| = 1$ and minimize $\|w\|$. \\

\[ r^t \left( \frac{w^T x + w_0}{\|w\|} \right) \geq \rho, \forall t \]
The solution, if it exists, is always at a saddle point of the Lagrangian

\[ L_p = \frac{1}{2} \|w\|^2 - \sum_{t=1}^{N} \alpha^t \left[ r^t \left( w^T x^t + w_0 \right) - 1 \right] \]

Unconstrained problem using Lagrange multipliers (+ numbers)
In the figure below we have illustrated an extreme value problem with constraints. The point A is the largest value of the function $z = f(x, y)$ while the point B is the largest value of the function under the constraint $g(x, y) = 0$. 
The method of Lagrange multipliers allows us to maximize or minimize functions with the constraint that we only consider points on a certain surface. To find critical points of a function \( f(x, y, z) \) on a level surface \( g(x, y, z) = C \) (or subject to the constraint \( g(x, y, z) = C \)), we must solve the following system of simultaneous equations:

\[
\nabla f(x, y, z) = \lambda \nabla g(x, y, z)
\]

\[g(x, y, z) = C\]

Remembering that \( \nabla f \) and \( \nabla g \) are vectors, we can write this as a collection of four equations in the four unknowns \( x, y, z, \) and \( \lambda \):

\[
\begin{align*}
f_x(x, y, z) &= \lambda g_x(x, y, z) \\
f_y(x, y, z) &= \lambda g_y(x, y, z) \\
f_z(x, y, z) &= \lambda g_z(x, y, z) \\
g(x, y, z) &= C
\end{align*}
\]

The variable \( \lambda \) is a dummy variable called a “Lagrange multiplier”; we only really care about the values of \( x, y, \) and \( z \).
The diagram shows a linear function \( f(x, y) = ax + by \) subject to a constraint \( x^2 + y^2 = c \). Here \( \nabla f = (a, b) \) is constant, \( \nabla g = (2x, 2y) \), and the constrained extrema of \( f \) occur at the points where \( (a, b) \) is perpendicular to the circle.
\[
\min \frac{1}{2} \|w\|^2 \text{ subject to } r^t(w^T x^t + w_0) \geq +1, \forall t
\]

\[
L_p = \frac{1}{2} \|w\|^2 - \sum_{t=1}^{N} \alpha^t [r^t(w^T x^t + w_0) - 1]
\]

\[
= \frac{1}{2} \|w\|^2 - \sum_{t=1}^{N} \alpha^t r^t(w^T x^t + w_0) + \sum_{t=1}^{N} \alpha^t
\]

\[
\frac{\partial L_p}{\partial w} = 0 \quad \Rightarrow \quad w = \sum_{t=1}^{N} \alpha^t r^t x^t
\]

\[
\frac{\partial L_p}{\partial w_0} = 0 \quad \Rightarrow \quad \sum_{t=1}^{N} \alpha^t r^t = 0
\]

Convex quadratic optimization problem can be solved using the dual form where we use these local minima constraints and maximize w.r.t \(\alpha^t\)s.
Problem: maximize

\[ f(x, y) = 6x + 8y \]

subject to

\[ g(x, y) = x^2 + y^2 - 1 \geq 0 \]

Using a Lagrange multiplier \( a \),

\[ \max_{xy} \min_{a \geq 0} f(x, y) + ag(x, y) \]

At optimum,

\[ 0 = \nabla f(x, y) + a \nabla g(x, y) = \begin{pmatrix} 6 \\ 8 \end{pmatrix} + 2a \begin{pmatrix} x \\ y \end{pmatrix} \]

from: http://math.oregonstate.edu/home/programs/undergrad/CalculusQuestStudyGuides/vcalc/lagrang/lagrang.html
\[ L_p = \frac{1}{2} \| w \|^2 - \sum_{t=1}^{N} \alpha^t \left[ r^t (w^T x^t + w_0) - 1 \right] \]
\[ \frac{\partial L_p}{\partial w} = 0 \Rightarrow w = \sum_{t=1}^{N} \alpha^t r^t x^t \]
\[ \frac{\partial L_p}{\partial w_0} = 0 \Rightarrow \sum_{t=1}^{N} \alpha^t r^t = 0 \]

\[ L_d = \frac{1}{2} (w^T w) - w^T \sum_t \alpha^t r^t x^t - w_0 \sum_t \alpha^t r^t + \sum_t \alpha^t \]
\[ = -\frac{1}{2} (w^T w) + \sum_t \alpha^t \]
\[ = -\frac{1}{2} \sum_t \sum_s \alpha^t \alpha^s r^t r^s (x^t)^T x^s + \sum_t \alpha^t \]

subject to \( \sum_t \alpha^t r^t = 0 \) and \( \alpha^t \geq 0, \forall t \)

• Maximize \( L_d \) with respect to \( \alpha^t \) only
• Quadratic programming problem
• Thanks to the convexity of the problem, optimal value of \( L_p = L_d \)
To every convex program corresponds a dual

Solving original (primal) is equivalent to solving dual
Support vectors for which
\[ r^T \left( w^T x^T + w_0 \right) = 1 \]
\[
L_d = \frac{1}{2} (w^T w) - w^T \sum_t \alpha^t r^t x^t - w_0 \sum_t \alpha^t r^t + \sum_t \alpha^t \\
= -\frac{1}{2} (w^T w) + \sum_t \alpha^t \\
= -\frac{1}{2} \sum_t \sum_s \alpha^t \alpha^s r^t r^s (x^t)^T x^s + \sum_t \alpha^t
\]

\text{Size of the dual depends on N and not on d}

• Maximize \( L_d \) with respect to \( \alpha^t \) only

subject to \( \sum_t \alpha^t r^t = 0 \) and \( \alpha^t \geq 0, \forall t \)

• Quadratic programming problem
• Thanks to the convexity of the problem, optimal value of \( L_p = L_d \)
Calculating the parameters $w$ and $w_0$

Note that:

- either the constraint is exactly satisfied (=1)
  (and $\alpha^t$ can be non-zero)
- or the constraint is clearly satisfied (> 1)
  (then $\alpha^t$ must be zero)

Once we solve for $\alpha^t$, we see that most of them are 0 and only a small number have $\alpha^t > 0$
- the corresponding $x^t$s are called the support vectors
Calculating the parameters $\mathbf{w}$ and $\mathbf{w}_0$

Once we have the Lagrange multipliers, we can compute $\mathbf{w}$ and $\mathbf{w}_0$:

$$\mathbf{w} = \sum_{t=1}^{N} \alpha^t \mathbf{r}^t \mathbf{x}^t = \sum_{t \in SV} \alpha^t \mathbf{r}^t \mathbf{x}^t$$

where $SV$ is the set of the Support Vectors.

$$\mathbf{w}_0 = \mathbf{r}^t - \mathbf{w}^T \mathbf{x}^t$$
We make decisions by comparing each query $x$ with only the support vectors

$$y = \text{sign}(w^T x + w_0) = \left( \sum_{t \in SV} \alpha^t r^t x^t \right) x + w_0$$

Choose class C1 if $+$, C2 if negative
Not-Linearily Separable Case
The non-separable case cannot find a feasible solution using the previous approach

- The objective function \( L_D \) grows arbitrarily large.

Relax the constraints, but only when necessary

- Introduce a further cost for this
Soft Margin Hyperplane

- **Not linearly separable**

\[
r^t \left( w^T x^t + w_0 \right) \geq 1 - \xi^t
\]

\[
\xi^t \geq 0
\]

Three cases (shown in fig):

Case 1: \( \xi^t = 0 \)

Case 2: \( \xi^t \geq 1 \)

Case 3: \( 0 \leq \xi^t < 1 \)
Soft Margin Hyperplane

- Define Soft error
  \[ \sum_t \xi_t \]

- New primal is
  \[ L_p = \frac{1}{2} \|w\|^2 + C \sum_t \xi_t - \sum_t \alpha^t \left[ r^t (w^T x^t + w_0) - 1 + \xi^t \right] - \sum_t u^t \xi^t \]

- Parameter C can be viewed as a way to control overfitting: it “trades off” the relative importance of maximizing the margin and fitting the training data.

Upper bound on the number of training errors

Lagrange multipliers to enforce positivity of \( \xi \)
New dual is the same as the old one

\[ L_d = -\frac{1}{2} \sum_t \sum_s \alpha^t \alpha^s r^t r^s (x^t)^T x^s + \sum_t \alpha^t \]

subject to

\[ \sum_t \alpha^t r^t = 0 \text{ and } 0 \leq \alpha^t \leq C, \forall t \]

As in the separable case, instances that are not support vectors vanish with their \( \alpha^t = 0 \) and the remaining define the boundary.
Kernel Functions in SVM
- We can handle the overfitting problem: even if we have lots of parameters, large margins make simple classifiers

- “All” that is left is efficiency

- Solution: kernel trick
Kernel Functions

- Instead of trying to fit a non-linear model, we can
  - map the problem to a new space through a non-linear transformation and
  - use a linear model in the new space

- Say we have the new space calculated by the basis functions $z = \phi(x)$ where $z_j = \phi_j(x)$, $j=1,...,k$

  d-dimensional $\mathbf{x}$ space $\rightarrow$ k-dimensional $\mathbf{z}$ space

$\phi(x) = [\phi_1(x) \ \phi_2(x) \ ... \ \phi_k(x)]$
\( \varphi : \mathbb{R}^2 \rightarrow \mathbb{R}^3 \)

\((x_1, x_2) \mapsto (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)\)
Kernel Functions

\[ g(x) = \sum_{k=1}^{\infty} w_k \varphi_k(x) + b \]

\[ g(x) = \sum_{k=0}^{\infty} w_k \varphi_k(x) \]

if we assume \( \varphi_0(x) = 1 \) for \( \forall x \)
Kernel Machines

- Preprocess input \( x \) by basis functions
  \[
  z = \phi(x) \quad g(z) = w^T z \\
  g(x) = w^T \phi(x)
  \]

- SVM solution: Find Kernel functions \( K(x,y) \) such that the inner product of basis functions are replaced by a Kernel function in the original input space

\[
w = \sum_t \alpha^t r^t z^t = \sum_t \alpha^t r^t \phi(x^t) \\
g(x) = w^T \phi(x) = \sum_t \alpha^t r^t \phi(x^t)^T \phi(x) \\
g(x) = \sum_t \alpha^t r^t K(x^t, x)
\]
Consider polynomials of degree $q$:

$$K(x, y) = (x^T y + 1)^q$$

$$K(x, y) = (x^T y + 1)^2$$

$$= (x_1 y_1 + x_2 y_2 + 1)^2$$

$$= 1 + 2x_1 y_1 + 2x_2 y_2 + 2x_1 x_2 y_1 y_2 + x_1^2 y_1^2 + x_2^2 y_2^2$$

$$\phi(x) = \begin{bmatrix} 1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2 \end{bmatrix}^T$$

(Cherkassky and Mulier, 1998)
\[ x = (x_1, x_2); \]
\[ z = (z_1, z_2); \]
\[
\langle x, z \rangle^2 = (x_1 z_1 + x_2 z_2)^2 =
\]
\[ = x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 z_1 x_2 z_2 =
\]
\[ = \langle (x_1^2, x_2^2, \sqrt{2} x_1 x_2), (z_1^2, z_2^2, \sqrt{2} z_1 z_2) \rangle =
\]
\[ = \langle \phi(x), \phi(z) \rangle \]

www.support-vector.net
Examples of Kernel Functions

- **Linear**: $K(x_i, x_j) = x_i^T x_j$
  - Mapping $\Phi$: $x \rightarrow \phi(x)$, where $\phi(x)$ is $x$ itself

- **Polynomial of power $p$**: $K(x_i, x_j) = (1 + x_i^T x_j)^p$
  - Mapping $\Phi$: $x \rightarrow \phi(x)$, where $\phi(x)$ has $\binom{d + p}{p}$ dimensions

- **Gaussian (radial-basis function)**: $K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$
  - Mapping $\Phi$: $x \rightarrow \phi(x)$, where $\phi(x)$ is infinite-dimensional: every point is mapped to a function (a Gaussian)

- Higher-dimensional space still has intrinsic dimensionality $d$, but linear separators in it correspond to non-linear separators in original space.
Typically $k$ is much larger than $d$, and possibly larger than $N$
- Using the dual where the complexity depends on $N$ rather than $k$ is advantageous

We use the soft margin hyperplane
- If $C$ is too large, too high a penalty for non-separable points (too many support vectors)
- If $C$ is too small, we may have underfitting

Decide by cross-validation
Other Kernel Functions

- Polynomials of degree $q$:
  $$K(x^t, x) = (x^T x^t)^q$$
  $$K(x^t, x) = (x^T x^t + 1)^q$$

- Radial-basis functions:
  $$K(x^t, x) = \exp\left[-\frac{\|x^t - x\|^2}{\sigma^2}\right]$$

- Sigmoidal functions such as:
  $$K(x^t, x) = \tanh(2x^T x^t + 1)$$
What Functions are Kernels? Advanced

- For some functions $K(x_i, x_j)$ checking that $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ can be cumbersome.

- Any function that satisfies some constraints called the Mercer conditions can be a Kernel function - (Cherkassky and Mulier, 1998)
  
  Every semi-positive definite symmetric function is a kernel

- Semi-positive definite symmetric functions correspond to a semi-positive definite symmetric Gram matrix:

  $\begin{pmatrix}
  K(x_1, x_1) & K(x_1, x_2) & K(x_1, x_3) & \ldots & K(x_1, x_n) \\
  K(x_2, x_1) & K(x_2, x_2) & K(x_2, x_3) & \ldots & K(x_2, x_n) \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  K(x_n, x_1) & K(x_n, x_2) & K(x_n, x_3) & \ldots & K(x_n, x_n)
  \end{pmatrix}$
Informally, kernel methods *implicitly* define the class of possible patterns by introducing a notion of similarity between data

- Choice of similarity -> Choice of relevant features

More formally, kernel methods exploit information about the inner products between data items

- Many standard algorithms can be rewritten so that they only require inner products between data (inputs)
- Kernel functions = inner products in some feature space (potentially very complex)
- If kernel given, no need to specify what features of the data are being used
- Kernel functions make it possible to use infinite dimensions
  - efficiently in time / space
String kernels

- For example, given two documents, $D_1$ and $D_2$, the number of words appearing in both may form a kernel.

- Define $\phi(D_1)$ as the M-dimensional binary vector where dimension $i$ is 1 if word $w_i$ appears in $D_1$; 0 otherwise.

- Then $\phi(D_1)^T \phi(D_2)$ indicates the number of shared words.

- If we define $K(D_1,D_2)$ as the number of shared words:
  - no need to preselect the M words
  - no need to create the bag-of-words model explicitly
  - M can be as large as we want
Projecting into Higher Dimensions

- Naïve application of this concept by simply projecting to a high-dimensional non-linear manifold has two major problems
  - **Statistical**: operation on high-dimensional spaces is ill-conditioned due to the "curse of dimensionality" and the subsequent risk of overfitting
  - **Computational**: working in high-dimensions requires higher computational power, which poses limits on the size of the problems that can be tackled

- **SVMs bypass these two problems in a robust and efficient manner**
  - First, generalization capabilities in the high-dimensional manifold are ensured by enforcing a **largest margin** classifier
    - Recall that generalization in SVMs is strictly a function of the margin (or the VC dimension), regardless of the dimensionality of the feature space
  - Second, projection onto a high-dimensional manifold is only **implicit**
    - Recall that the SVM solution depends only on the dot product $\langle x_i, x_j \rangle$ between training examples
    - Therefore, operations in high dimensional space $\phi(x)$ do not have to be performed explicitly if we find a function $K(x_i, x_j)$ such that $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$
    - $K(x_i, x_j)$ is called a **kernel** function in SVM terminology
SVM Applications

- Cortes and Vapnik 1995:
  - Handwritten digit classification
  - 16x16 bitmaps -> 256 dimensions
  - Polynomial kernel where $q=3$ -> feature space with $10^6$ dimensions
  - No overfitting on a training set of 7300 instances
  - Average of 148 support vectors over different training sets

Expected test error rate:

$$\text{Exp}_N[P(\text{error})] = \text{Exp}_N[\#\text{support vectors}] / N$$

(= 0.02 for the above example)
SVMs were originally proposed by Boser, Guyon and Vapnik in 1992 and gained increasing popularity in late 1990s.

SVMs represent a general methodology for many PR problems: classification, regression, feature extraction, clustering, novelty detection, etc.

SVMs can be applied to complex data types beyond feature vectors (e.g. graphs, sequences, relational data) by designing kernel functions for such data.

SVM techniques have been extended to a number of tasks such as regression [Vapnik et al. ’97], principal component analysis [Schölkopf et al. ’99], etc.

Most popular optimization algorithms for SVMs use decomposition to hill-climb over a subset of $a_i$’s at a time, e.g. SMO [Platt ’99] and [Joachims ’99]
Advantages of SVMs

- There are no problems with local minima, because the solution is a Quadratic Programming problem with a global minimum.
- The optimal solution can be found in polynomial time.
- There are few model parameters to select: the penalty term C, the kernel function and parameters (e.g., spread $\sigma$ in the case of RBF kernels).
- The final results are stable and repeatable (e.g., no random initial weights).
- The SVM solution is sparse; it only involves the support vectors.
- SVMs rely on elegant and principled learning methods.
- SVMs provide a method to control complexity independently of dimensionality.
- SVMs have been shown (theoretically and empirically) to have excellent generalization capabilities.
Challenges

- Can the kernel functions be selected in a principled manner?
- SVMs still require selection of a few parameters, typically through cross-validation
- How does one incorporate domain knowledge?
  - Currently this is performed through the selection of the kernel and the introduction of “artificial” examples
- How interpretable are the results provided by an SVM?
- What is the optimal data representation for SVM? What is the effect of feature weighting? How does an SVM handle categorical or missing features?
- Do SVMs always perform best? Can they beat a hand-crafted solution for a particular problem?
- Do SVMs eliminate the model selection problem?
More explanations or demonstrations can be found at:

- Haykin Chp. 6 pp. 318-339
- Burges tutorial (under/reading/)
- http://www.dtreg.com/svm.htm

Software

- **SVMlight**, by Joachims, is one of the most widely used SVM classification and regression package. Distributed as C++ source and binaries for Linux, Windows, Cygwin, and Solaris. Kernels: polynomial, radial basis function, and neural (tanh).

- **LibSVM** [http://www.csie.ntu.edu.tw/~cjlin/libsvm/](http://www.csie.ntu.edu.tw/~cjlin/libsvm/) LIBSVM (Library for Support Vector Machines), is developed by Chang and Lin; also widely used. Developed in C++ and Java, it supports also multi-class classification, weighted SVM for unbalanced data, cross-validation and automatic model selection. It has interfaces for Python, R, Splus, MATLAB, Perl, Ruby, and LabVIEW. Kernels: linear, polynomial, radial basis function, and neural (tanh).

Applet to play with:

This applet demonstrates SVM (Support Vector Machine) data classification. It shows a graph with two classes of data points, separated by a decision boundary. The SVM applet allows users to select predefined samples, set kernel options, and view detailed SVM options like stopping criteria (epsilon), coefficient of the error term (C), use shrinking, and cache in megabytes. The SVM results display iterations, rho values, probability of class A (probA), probability of class B (probB), number of support vectors, and other parameters. Developed for EE-583 Pattern Recognition, developed by Hakan Serçe, 2005.
SVM Applet...

Developed for:
EE-583 Pattern Recognition

Developed by:
Hakan Serçe, 2005

This applet demonstrates SVM (Support Vector Machine)