Learning with Kernels

Bernhard Schölkopf
Max-Planck-Institut für biologische Kybernetik
72076 Tübingen, Germany
bernhard.schoelkopf@tuebingen.mpg.de

Learning and Similarity: some Informal Thoughts

- input/output sets $\mathcal{X}, \mathcal{Y}$
- training set $(x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{X} \times \mathcal{Y}$
- "generalization": given a previously unseen $x \in \mathcal{X}$, find a suitable $y \in \mathcal{Y}$
- (x, y) should be "similar" to $(x_1, y_1), \ldots, (x_m, y_m)$
- how to measure similarity?
  - for outputs: *loss function* (e.g., for $\mathcal{Y} = \{\pm 1\}$, zero-one loss)
  - for inputs: *kernel*

Roadmap

1. Introduction: similarity measures and machine learning
2. Elements of Statistical Learning Theory
3. Kernels
4. Support Vector classification
5. Support Vector regression

Similarity of Inputs

- symmetric function

$$\begin{aligned} k : \mathcal{X} \times \mathcal{X} &\to \mathbb{R} \\ (x, x') &\mapsto k(x, x') \end{aligned}$$

- for example, if $\mathcal{X} = \mathbb{R}^N$: canonical dot product

$$k(x, x') = \sum_{i=1}^{N} [x]_i [x']_i$$

- if $\mathcal{X}$ is not a dot product space: assume that k has a representation as a dot product in a linear space $\mathcal{H}$, i.e., there exists a map $\Phi : \mathcal{X} \to \mathcal{H}$ such that

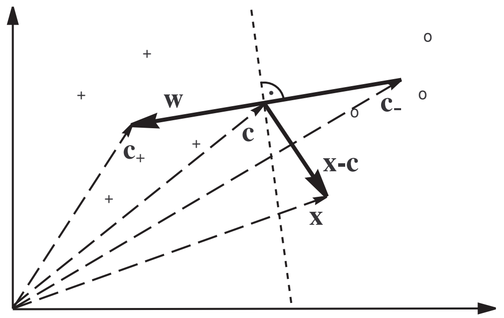
$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle .$$

- in that case, we can think of the patterns as $\Phi(x), \Phi(x')$, and carry out geometric algorithms in the dot product space ("feature space") $\mathcal{H}$.

An Example of a Kernel Algorithm

Idea: classify points $\mathbf{x} := \Phi(x)$ in feature space according to which of the two class means is closer.

$$\mathbf{c}_+ := \frac{1}{m_1} \sum_{y_i=1} \Phi(x_i), \quad \mathbf{c}_- := \frac{1}{m_2} \sum_{y_i=-1} \Phi(x_i)$$



Compute the sign of the dot product between $\mathbf{w} := \mathbf{c}_+ - \mathbf{c}_-$ and $\mathbf{x} - \mathbf{c}$.

An Example of a Kernel Algorithm, ctd.

- Demo
- Exercise: derive the Parzen windows classifier by computing the distance criterion directly

An Example of a Kernel Algorithm, ctd. [50]

$$\begin{aligned} f(x) &= \operatorname{sgn}\left(\frac{1}{m_1} \sum_{\{i:y_i=+1\}} \langle \Phi(x), \Phi(x_i) \rangle - \frac{1}{m_2} \sum_{\{i:y_i=-1\}} \langle \Phi(x), \Phi(x_i) \rangle + b \right) \\ &= \operatorname{sgn}\left(\frac{1}{m_1} \sum_{\{i:y_i=+1\}} k(x, x_i) - \frac{1}{m_2} \sum_{\{i:y_i=-1\}} k(x, x_i) + b \right) \end{aligned}$$

where

$$b = \frac{1}{2} \left(\frac{1}{m_2^2} \sum_{\{(i,j):y_i=y_j=-1\}} k(x_i, x_j) - \frac{1}{m_1^2} \sum_{\{(i,j):y_i=y_j=+1\}} k(x_i, x_j) \right).$$

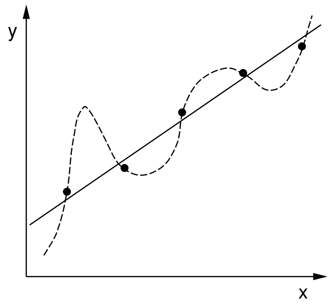
- provides a geometric interpretation of Parzen windows
- the decision function is a hyperplane. Will it generalize well?

Statistical Learning Theory

1. started by Vapnik and Chervonenkis in the Sixties
2. model: we observe data generated by an unknown stochastic regularity
3. *learning* = extraction of the regularity from the data
4. the analysis of the learning problem leads to notions of *capacity* of the function classes that a learning machine can implement.
5. *support vector machines* use a particular type of function class: classifiers with large "margins" in a feature space induced by a *kernel*.

[62, 63]

Example: Regression Estimation



- *Data:* input-output pairs $(x_i, y_i) \in \mathbb{R} \times \mathbb{R}$
- *Regularity:* $(x_1, y_1), \ldots (x_m, y_m)$ drawn from $\mathrm{P}(x, y)$
- *Learning:* choose a function $f : \mathbb{R} \to \mathbb{R}$ such that the error, averaged over P, is minimized.
- *Problem:* P is unknown, so the average cannot be computed — need an "*induction* principle"

Pattern Recognition

Learn $f : \mathcal{X} \to \{\pm 1\}$ from examples

$(x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{X} \times \{\pm 1\}$, generated i.i.d. from $\mathrm{P}(x, y)$,

such that the expected misclassification error on a test set, also drawn from $\mathrm{P}(x, y)$,

$$R[f] = \int \frac{1}{2} |f(x) - y| \, d\mathrm{P}(x, y),$$

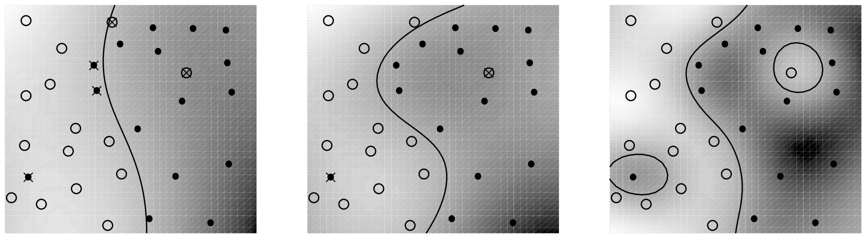
is minimal *(Risk Minimization (RM))*.

Problem: P is unknown. $\longrightarrow$ need an *induction principle*.

Empirical risk minimization (ERM): replace the average over $\mathrm{P}(x, y)$ by an average over the training sample, i.e. minimize the training error

$$R_{\mathrm{emp}}[f] = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} |f(x_i) - y_i|$$

Example: Pattern Recognition



Risk minimization

- Regression estimation. RM: minimize

$$R[f] = \int (f(x) - y)^2 \, d\mathrm{P}(x, y)$$

— leads to the *regression* $y(x) = \int y \, d\mathrm{P}(y|x)$.

ERM gives least mean squares: minimize

$$\sum_i (f(x_i) - y_i)^2$$

- Density estimation. RM: minimize

$$R[f] = \int (-\log p(x)) \, d\mathrm{P}(x)$$

ERM gives maximum likelihood estimation: maximize

$$\sum_i \log p(x_i) = \log(\prod_i p(x_i))$$

Convergence of Means to Expectations

Law of large numbers:

$$R_{\text{emp}}[f] \to R[f]$$

as $m \to \infty$.

Does this imply that empirical risk minimization will give us the optimal result in the limit of infinite sample size (*"consistency"* of empirical risk minimization)?

No.
Need a *uniform* version of the law of large numbers. Uniform over *all functions that the learning machine can implement.*

The Importance of the Set of Functions

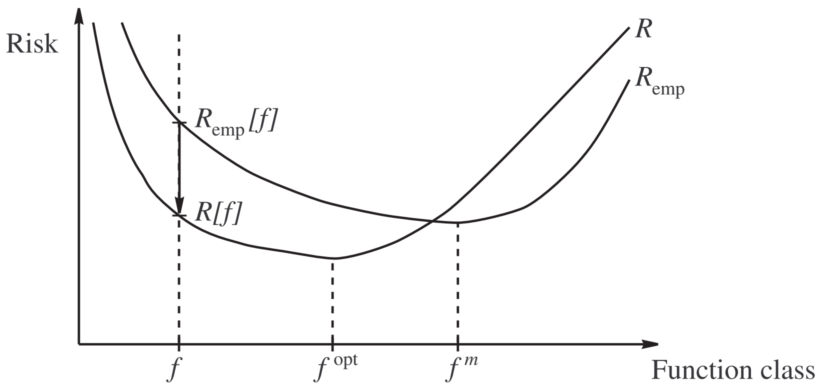
What about allowing *all* functions from $\mathcal{X}$ to $\{\pm 1\}$?

Training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m) \in \mathcal{X} \times \{\pm 1\}$
Test patterns $\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_{\bar{m}} \in \mathcal{X}$,
such that $\{\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_{\bar{m}}\} \cap \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} = \{\}$.

For any f there exists f^* s.t.:
1. $f^*(\mathbf{x}_i) = f(\mathbf{x}_i)$ for all i
2. $f^*(\bar{\mathbf{x}}_j) \neq f(\bar{\mathbf{x}}_j)$ for all j.

Based on the training set alone, there is *no* means of choosing which one is better. On the test set, however, they give *opposite* results. There is 'no free lunch' [28, 72].
$\longrightarrow$ a restriction must be placed on the *functions* that we allow

Consistency and Uniform Convergence



Restricting the Class of Functions

Two views:

1. Statistical Learning (VC) Theory: take into account the *capacity* of the class of functions that the learning machine can implement

2. The Bayesian Way: place *Prior distributions* $\mathrm{P}(f)$ over the class of functions

Detailed Analysis

- loss $\xi_i := \frac{1}{2}|f(x_i) - y_i|$ in $\{0, 1\}$
- the ξ_i are independent Bernoulli trials
- empirical mean $\frac{1}{m}\sum_{i=1}^m \xi_i$ (by def: equals $R_{\text{emp}}[f]$)
- expected value $\mathbf{E}[\xi]$ (equals $R[f]$)

Chernoff's Bound

$$\mathrm{P}\left\{\left|\frac{1}{m}\sum_{i=1}^m \xi_i - \mathbf{E}[\xi]\right| \geq \epsilon\right\} \leq 2\exp(-2m\epsilon^2)$$

- here, P refers to the probability of getting a sample $\xi_1, \ldots, \xi_m$ with the property $\left|\frac{1}{m}\sum_{i=1}^m \xi_i - \mathbf{E}[\xi]\right| \geq \epsilon$ (is a product measure)

Useful corollary: Given a $2m$-sample of Bernoulli trials, we have

$$\mathrm{P}\left\{\left|\frac{1}{m}\sum_{i=1}^m \xi_i - \frac{1}{m}\sum_{i=m+1}^{2m} \xi_i\right| \geq \epsilon\right\} \leq 4\exp\left(-\frac{m\epsilon^2}{2}\right).$$

Chernoff's Bound, II

Translate this back into machine learning terminology: the probability of obtaining an m-sample where the training error and test error differ by more than $\epsilon > 0$ is bounded by

$$\mathrm{P}\left\{\left|R_{\text{emp}}[f] - R[f]\right| \geq \epsilon\right\} \leq 2\exp(-2m\epsilon^2).$$

- refers to one fixed f
- not allowed to look at the data before choosing f, hence not suitable as a bound on the test error of a learning algorithm using empirical risk minimization

Two Observations

- denote the minimizer of R by f^{opt}. Then
$$R[f] - R[f^{\text{opt}}] \geq 0$$
for all $f \in \mathcal{F}$.
- denote the minimizer of R_{emp} by f^m. Then
$$R_{\text{emp}}[f] - R_{\text{emp}}[f^m] \geq 0$$
for all $f \in \mathcal{F}$.
- In particular, we have
$$R[f^m] - R[f^{\text{opt}}] \geq 0$$
and
$$R_{\text{emp}}[f^{\text{opt}}] - R_{\text{emp}}[f^m] \geq 0.$$

The sum of these two inequalities satisfies

$$\begin{aligned} 0 &\leq R[f^m] - R[f^{\text{opt}}] + R_{\text{emp}}[f^{\text{opt}}] - R_{\text{emp}}[f^m] \\ &= R[f^m] - R_{\text{emp}}[f^m] + R_{\text{emp}}[f^{\text{opt}}] - R[f^{\text{opt}}] \\ &\leq \sup_{f \in \mathcal{F}} (R[f] - R_{\text{emp}}[f]) + (R_{\text{emp}}[f^{\text{opt}}] - R[f^{\text{opt}}]). \end{aligned}$$

- second half of RHS: f^{opt} is fixed (independent of training sample), hence by Chernoff: for all $\epsilon > 0$,

  $$\lim_{m \to \infty} \mathrm{P}\{|R_{\text{emp}}[f^{\text{opt}}] - R[f^{\text{opt}}]| > \epsilon\} = 0$$

  ("convergence in probability")

- If the first half of RHS also converges to zero (in probability), i.e.,

  $$\lim_{m \to \infty} \mathrm{P}\{\sup_{f \in \mathcal{F}} (R[f] - R_{\text{emp}}[f]) > \epsilon\} = 0,$$

  for all $\epsilon > 0$, then

  $$\begin{aligned} 0 \leq\; & R[f^m] - R[f^{\text{opt}}] \to 0 \\ 0 \leq\; & R_{\text{emp}}[f^{\text{opt}}] - R_{\text{emp}}[f^m] \to 0 \end{aligned}$$

  in probability — in this case, empirical risk minimization can be seen to be *consistent*.

Uniform Convergence (Vapnik & Chervonenkis)

Necessary and sufficient conditions for nontrivial consistency of empirical risk minimization (ERM):
One-sided convergence, uniformly over all functions that can be implemented by the learning machine.

$$\lim_{m \to \infty} P\{\sup_{f \in \mathcal{F}} (R[f] - R_{\text{emp}}[f]) > \epsilon\} = 0$$

for all $\epsilon > 0$.

- note that this takes into account the whole set of functions that can be implemented by the learning machine
- this is hard to check for a learning machine

Are there properties of learning machines ($\equiv$ sets of functions) which ensure uniform convergence of risk?

How to Prove a VC Bound

Take a closer look at $\mathrm{P}\{\sup_{f \in \mathcal{F}}(R[f] - R_{\text{emp}}[f]) > \epsilon\}$.
Plan:

- if the function class $\mathcal{F}$ contains only one function, then Chernoff's bound suffices:

  $$\mathrm{P}\{\sup_{f \in \mathcal{F}} (R[f] - R_{\text{emp}}[f]) > \epsilon\} \leq 2\exp(-2m\epsilon^2).$$

- if there are finitely many functions, we use the 'union bound'
- even if there are infinitely many, then *on any finite sample* there are effectively only finitely many (use *symmetrization* and *capacity concepts*)

The Case of Two Functions

Suppose $\mathcal{F} = \{f_1, f_2\}$. Rewrite

$$\mathrm{P}\{\sup_{f\in\mathcal{F}}(R[f] - R_{\mathrm{emp}}[f]) > \epsilon\} = \mathrm{P}(C^1_\epsilon \cup C^2_\epsilon),$$

where

$$C^i_\epsilon := \{(x_1, y_1), \ldots, (x_m, y_m) \mid (R[f_i] - R_{\mathrm{emp}}[f_i]) > \epsilon\}$$

denotes the event that the risks of f_i differ by more than ϵ.
The RHS equals

$$\begin{aligned}\mathrm{P}(C^1_\epsilon \cup C^2_\epsilon) &= \mathrm{P}(C^1_\epsilon) + \mathrm{P}(C^2_\epsilon) - \mathrm{P}(C^1_\epsilon \cap C^2_\epsilon)\\ &\leq \mathrm{P}(C^1_\epsilon) + \mathrm{P}(C^2_\epsilon).\end{aligned}$$

Hence by Chernoff's bound

$$\begin{aligned}\mathrm{P}\{\sup_{f\in\mathcal{F}}(R[f] - R_{\mathrm{emp}}[f]) > \epsilon\} &\leq \mathrm{P}(C^1_\epsilon) + \mathrm{P}(C^2_\epsilon)\\ &\leq 2 \cdot 2\exp(-2m\epsilon^2).\end{aligned}$$

The Union Bound

Similarly, if $\mathcal{F} = \{f_1, \ldots, f_n\}$, we have

$$\mathrm{P}\{\sup_{f\in\mathcal{F}}(R[f] - R_{\mathrm{emp}}[f]) > \epsilon\} = \mathrm{P}(C^1_\epsilon \cup \cdots \cup C^n_\epsilon),$$

and

$$\mathrm{P}(C^1_\epsilon \cup \cdots \cup C^n_\epsilon) \leq \sum_{i=1}^n \mathrm{P}(C^i_\epsilon).$$

Use Chernoff for each summand, to get an extra factor n in the bound.

Note: this becomes an equality if and only if all the events involved are *disjoint*.

Infinite Function Classes

- Note: empirical risk only refers to m points. On these points, the functions of $\mathcal{F}$ can take at most 2^m values
- for R_{emp}, the function class thus "looks" finite
- how about R?
- need to use a trick

Symmetrization

Lemma 1 (Vapnik & Chervonenkis (e.g., [59, 21])) *For $m\epsilon^2 \geq 2$ we have*

$$\mathrm{P}\{\sup_{f\in\mathcal{F}}(R[f]-R_{\mathrm{emp}}[f]) > \epsilon\} \leq 2\mathrm{P}\{\sup_{f\in\mathcal{F}}(R_{\mathrm{emp}}[f]-R'_{\mathrm{emp}}[f]) > \epsilon/2\}$$

Here, the first P *refers to the distribution of iid samples of size m, while the second one refers to iid samples of size $2m$. In the latter case, R_{emp} measures the loss on the first half of the sample, and R'_{emp} on the second half.*

Shattering Coefficient

- Hence, we only need to consider the maximum size of $\mathcal{F}$ on $2m$ points. Call it $\mathcal{N}(\mathcal{F}, 2m)$.
- $\mathcal{N}(\mathcal{F}, 2m)$ = max. number of different outputs $(y_1, \ldots, y_{2m})$ that the function class can generate on $2m$ points — in other words, the max. number of different ways the function class can separate $2m$ points into two classes.
- $\mathcal{N}(\mathcal{F}, 2m) \leq 2^{2m}$
- if $\mathcal{N}(\mathcal{F}, 2m) = 2^{2m}$, then the function class is said to *shatter* $2m$ points.

ctd.

Use Chernoff's bound for each term:*

$$\mathrm{P}\left\{\frac{1}{m}\sum_{i=1}^{m}\xi_i - \frac{1}{m}\sum_{i=m+1}^{2m}\xi_i \geq \epsilon\right\} \leq 2\exp\left(-\frac{m\epsilon^2}{2}\right).$$

This yields

$$\mathrm{P}\{\sup_{f\in\mathcal{F}}(R[f] - R_{\mathrm{emp}}[f]) > \epsilon\} \leq 4\mathcal{N}(\mathcal{F}, 2m)\exp\left(-\frac{m\epsilon^2}{8}\right).$$

- provided that $\mathcal{N}(\mathcal{F}, 2m)$ does not grow exponentially in m, this is nontrivial
- such bounds are called *VC type inequalities*
- two types of randomness: (1) the P refers to the drawing of the training examples, and (2) $R[f]$ is an expectation over the drawing of test examples.

* A rigorous treatment would need to use a second randomization over permutations of the $2m$-sample, see [50].

Putting Everything Together

We now use (1) symmetrization, (2) the shattering coefficient, and (3) the union bound, to get

$$\begin{aligned}
& \mathrm{P}\{\sup_{f\in\mathcal{F}}(R[f] - R_{\mathrm{emp}}[f]) > \epsilon\} \\
\leq\ & 2\mathrm{P}\{\sup_{f\in\mathcal{F}}(R_{\mathrm{emp}}[f] - R'_{\mathrm{emp}}[f]) > \epsilon/2\} \\
=\ & 2\mathrm{P}\{(R_{\mathrm{emp}}[f_1] - R'_{\mathrm{emp}}[f_1]) > \epsilon/2 \vee \ldots \vee (R_{\mathrm{emp}}[f_{\mathcal{N}(\mathcal{F},2m)}] - R'_{\mathrm{emp}}[f_{\mathcal{N}(\mathcal{F},2m)}]) > \epsilon/2\} \\
\leq\ & \sum_{n=1}^{\mathcal{N}(\mathcal{F},2m)} 2\mathrm{P}\{(R_{\mathrm{emp}}[f_n] - R'_{\mathrm{emp}}[f_n]) > \epsilon/2\}.
\end{aligned}$$

Confidence Intervals

Rewrite the bound: specify the probability with which we want R to be close to R_{emp}, and solve for ϵ:

With a probability of at least $1 - \delta$,

$$R[f] \leq R_{\mathrm{emp}}[f] + \sqrt{\frac{8}{m}\left(\ln(\mathcal{N}(\mathcal{F}, 2m)) + \ln\frac{4}{\delta}\right)}.$$

This bound holds independent of f; in particular, it holds for the function f^m minimizing the empirical risk.

Discussion

- tighter bounds are available (better constants etc.)
- cannot minimize the bound over f
- other capacity concepts can be used

Further PR Capacity Concepts

- exchange '$\mathbf{E}$' and 'ln': *annealed entropy*.

  $H_{\mathcal{F}}^{\text{ann}}(m)/m \to 0 \iff$ exponentially fast uniform convergence

- take 'max' instead of 'E': *growth function*.
  Note that $G_{\mathcal{F}}(m) = \ln \mathcal{N}(\mathcal{F}, m)$.

  $G_{\mathcal{F}}(m)/m \to 0 \iff$ exponential convergence for all underlying distributions P.

  $G_{\mathcal{F}}(m) = m \cdot \ln(2)$ for all $m \iff$ for any m, all loss vectors can be generated, i.e., the m points can be chosen such that by using functions of the learning machine, they can be separated in all 2^m possible ways (*shattered*).

VC Entropy

On an example $(\mathbf{x}, y)$, f causes a loss

$$c(x, y, f(x)) := \frac{1}{2}|f(x) - y| \in \{0, 1\}.$$

For a larger sample $(x_1, y_1) \ldots, (x_m, y_m)$, the different functions $f \in \mathcal{F}$ lead to a *set* of loss vectors

$$\boldsymbol{\xi}_f = (c(x_1, y_1, f(x_1)), \ldots, c(x_m, y_m, f(x_m))),$$

whose cardinality we denote by

$$\mathcal{N}\left(\mathcal{F}, (x_1, y_1) \ldots, (x_m, y_m)\right).$$

The *VC entropy* is defined as

$$H_{\mathcal{F}}(m) = \mathbf{E}\left[\ln \mathcal{N}\left(\mathcal{F}, (x_1, y_1) \ldots, (x_m, y_m)\right)\right],$$

where the expectation is taken over the random generation of the m-sample $(x_1, y_1) \ldots, (x_m, y_m)$ from P.

$H_{\mathcal{F}}(m)/m \to 0 \iff$ uniform convergence of risks (hence consistency)

Structure of the Growth Function

Either $G_{\mathcal{F}}(m) = m \cdot \ln(2)$ for all $m \in \mathbb{N}$

Or there exists some *maximal* m for which the above is possible. Call this number the *VC-dimension*, and denote it by h. For $m > h$,

$$G_{\mathcal{F}}(m) \leq h\left(\ln \frac{m}{h} + 1\right).$$

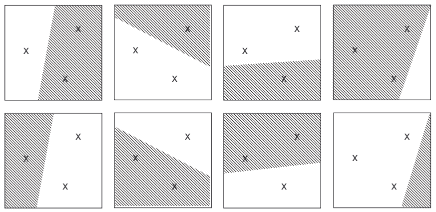
Nothing "in between" linear growth and logarithmic growth is possible.

VC-Dimension: Example

Half-spaces in $\mathbb{R}^2$:

$$f(x, y) = \operatorname{sgn}(a + bx + cy), \quad \text{with parameters } a, b, c \in \mathbb{R}$$

- Clearly, we can shatter three non-collinear points.
- But we can never shatter four points.
- Hence the VC dimension is $h = 3$ (in this case, equal to the number of parameters)



Examples of Induction Principles

- *Empirical risk minimization:* minimize

$$R_{\text{emp}}[f] = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} |f(\mathbf{x}_i) - y_i|$$

- *Minimum description length:* minimize some measure of the description length of the sequence $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ by a function f.
- *Structural risk minimization (SRM) (Vapnik, 1979):* minimize the RHS of

$$R[f] \leq R_{\text{emp}}[f] + \phi\left(\frac{h}{m}\right).$$

To this end, introduce a structure on $\mathcal{F}$.

Learning machine $\equiv$ a set of functions and an induction principle

A Typical Bound for Pattern Recognition

For any $f \in \mathcal{F}$ and $m > h$, with a probability of at least $1 - \delta$,

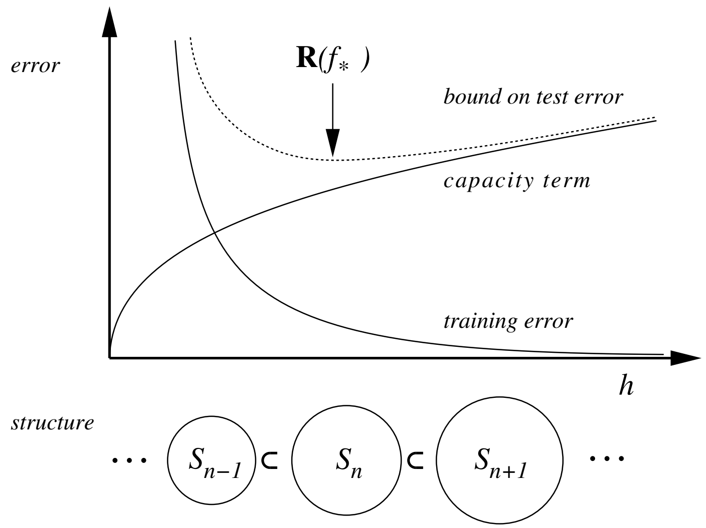
$$R[f] \leq R_{\text{emp}}[f] + \phi\left(\frac{h}{m}, \frac{\log(\delta)}{m}\right)$$

holds, where the *confidence term* ϕ is defined as

$$\phi\left(\frac{h}{m}, \frac{\log(\delta)}{m}\right) = \sqrt{\frac{h\left(\log \frac{2m}{h} + 1\right) - \log(\delta/4)}{m}}.$$

- does this mean, that we can learn *any*thing?
- The study of the consistency of ERM has thus led to concepts and results which lets us formulate a better induction principle: we can use this bound to get a low risk!
- in practice: use as a guideline for designing algorithms

SRM: The Picture



Finding a Good Function Class

- recall: separating hyperplanes in $\mathbb{R}^2$ have a VC dimension of 3.
- more generally: separating hyperplanes in $\mathbb{R}^N$ have a VC dimension of $N+1$.
- hence: separating hyperplanes in high-dimensional feature spaces have extremely large VC dimension, and may not generalize well
- however, *margin* hyperplanes can still have a small VC dimension

Canonical Hyperplanes

Note: if $c \neq 0$, then

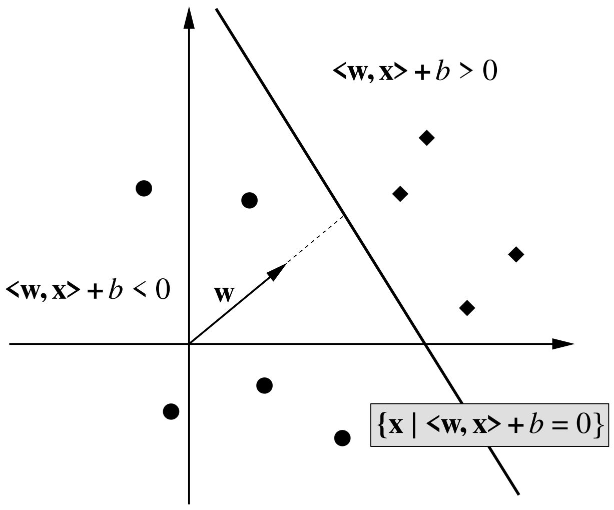
$$\{\mathbf{x} | \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\} = \{\mathbf{x} | \langle c\mathbf{w}, \mathbf{x} \rangle + cb = 0\}.$$

Hence $(c\mathbf{w}, cb)$ describes the same hyperplane as $(\mathbf{w}, b)$.

Definition: The hyperplane is in *canonical* form w.r.t. $X^* = \{\mathbf{x}_1, \ldots, \mathbf{x}_r\}$ if $\min_{\mathbf{x}_i \in X} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b| = 1$.

Note that for canonical hyperplanes, the distance of the closest point to the hyperplane ("margin") is $1/\|\mathbf{w}\|$:
$\min_{\mathbf{x}_i \in X} \left| \left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, \mathbf{x}_i \right\rangle + \frac{b}{\|\mathbf{w}\|} \right| = \frac{1}{\|\mathbf{w}\|}$.

Separating Hyperplane



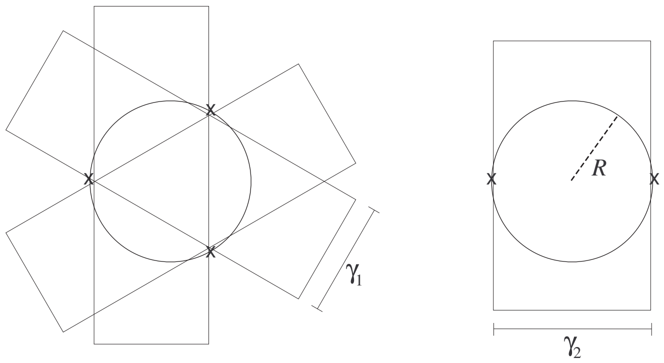
Theorem 2 (Vapnik [59]) *Consider hyperplanes $\langle \mathbf{w}, \mathbf{x} \rangle = 0$ where $\mathbf{w}$ is normalized such that they are in canonical form w.r.t. a set of points $X^* = \{\mathbf{x}_1, \ldots, \mathbf{x}_r\}$, i.e.,*

$$\min_{i=1,\ldots,r} |\langle \mathbf{w}, \mathbf{x}_i \rangle| = 1.$$

The set of decision functions $f_{\mathbf{w}}(\mathbf{x}) = \operatorname{sgn} \langle \mathbf{x}, \mathbf{w} \rangle$ defined on X^* and satisfying the constraint $\|\mathbf{w}\| \leq \Lambda$ has a VC dimension satisfying

$$h \leq R^2 \Lambda^2.$$

Here, R is the radius of the smallest sphere around the origin containing X^*.



recall $\gamma > 2/\Lambda$

Proof Strategy (Gurvits, 1997)

Assume that $\mathbf{x}_1, \ldots, \mathbf{x}_r$ are shattered by canonical hyperplanes with $\|\mathbf{w}\| \leq \Lambda$, i.e., for all $y_1, \ldots, y_r \in \{\pm 1\}$, there exists a $\mathbf{w}$ such that

$$y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1 \quad \text{for all } i = 1, \ldots, r. \tag{1}$$

Two steps:

- prove that the more points we want to shatter (1), the larger $\|\sum_{i=1}^r y_i \mathbf{x}_i\|$ must be
- upper bound the size of $\|\sum_{i=1}^r y_i \mathbf{x}_i\|$ in terms of R

Combining the two tells us how many points we can at most shatter.

Part I

Summing (1) over $i = 1, \ldots, r$ yields

$$\left\langle \mathbf{w}, \left(\sum_{i=1}^r y_i \mathbf{x}_i \right) \right\rangle \geq r.$$

By the Cauchy-Schwarz inequality, on the other hand, we have

$$\left\langle \mathbf{w}, \left(\sum_{i=1}^r y_i \mathbf{x}_i \right) \right\rangle \leq \|\mathbf{w}\| \left\| \sum_{i=1}^r y_i \mathbf{x}_i \right\| \leq \Lambda \left\| \sum_{i=1}^r y_i \mathbf{x}_i \right\|.$$

Combine both:

$$\frac{r}{\Lambda} \leq \left\| \sum_{i=1}^r y_i \mathbf{x}_i \right\|. \tag{2}$$

Part II

Consider independent random labels $y_i \in \{\pm 1\}$, uniformly distributed (*Rademacher variables*).

$$\begin{aligned}
\mathbf{E}\left[\left\| \sum_{i=1}^r y_i \mathbf{x}_i \right\|^2 \right] &= \sum_{i=1}^r \mathbf{E}\left[\left\langle y_i \mathbf{x}_i, \sum_{j=1}^r y_j \mathbf{x}_j \right\rangle \right] \\
&= \sum_{i=1}^r \mathbf{E}\left[\left\langle y_i \mathbf{x}_i, \left(\left(\sum_{j \neq i} y_j \mathbf{x}_j \right) + y_i \mathbf{x}_i \right) \right\rangle \right] \\
&= \sum_{i=1}^r \left(\left(\sum_{j \neq i} \mathbf{E}\left[\langle y_i \mathbf{x}_i, y_j \mathbf{x}_j \rangle \right] \right) + \mathbf{E}\left[\langle y_i \mathbf{x}_i, y_i \mathbf{x}_i \rangle \right] \right) \\
&= \sum_{i=1}^r \mathbf{E}\left[\|y_i \mathbf{x}_i\|^2 \right] = \sum_{i=1}^r \|\mathbf{x}_i\|^2
\end{aligned}$$

Part II, ctd.

Since $\|\mathbf{x}_i\| \leq R$, we get

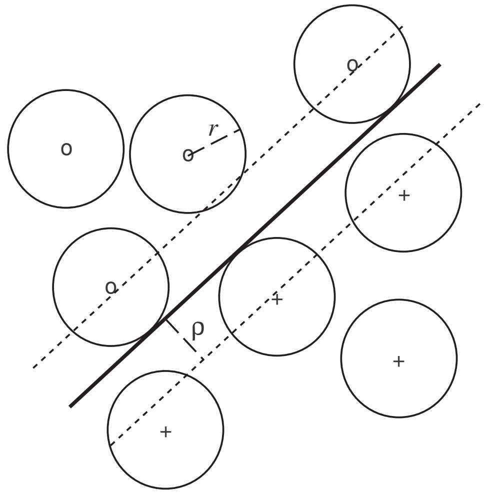
$$\mathbf{E}\left[\left\|\sum_{i=1}^{r} y_i \mathbf{x}_i\right\|^2\right] \leq r R^2.$$

- This holds for the *expectation* over the random choices of the labels, hence there must be at least one set of labels for which it also holds true. Use this set.

Hence

$$\left\|\sum_{i=1}^{r} y_i \mathbf{x}_i\right\|^2 \leq r R^2.$$

Pattern Noise as Maximum Margin Regularization



Part I and II Combined

Part I: $\left(\frac{r}{\Lambda}\right)^2 \leq \|\sum_{i=1}^{r} y_i \mathbf{x}_i\|^2$

Part II: $\|\sum_{i=1}^{r} y_i \mathbf{x}_i\|^2 \leq r R^2$

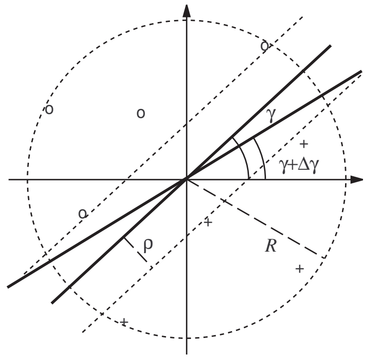
Hence

$$\frac{r^2}{\Lambda^2} \leq r R^2,$$

i.e.,

$$r \leq R^2 \Lambda^2.$$

Maximum Margin vs. MDL — 2D Case



Can perturb γ by $\Delta\gamma$ with $|\Delta\gamma| < \arcsin\frac{\rho}{R}$ and still correctly separate the data.
Hence only need to store γ with accuracy $\Delta\gamma$ [50, 65].

Kernels and Feature Spaces

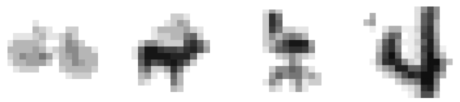
Preprocess the data with

$$\begin{aligned} \Phi : \mathcal{X} &\to \mathcal{H} \\ x &\mapsto \Phi(x), \end{aligned}$$

where $\mathcal{H}$ is a dot product space, and learn the mapping from $\Phi(x)$ to y.

- usually, $\dim(\mathcal{X}) \ll \dim(\mathcal{H})$
- "Curse of Dimensionality"?
- crucial issue: *capacity*, not *dimensionality*

General Product Feature Space



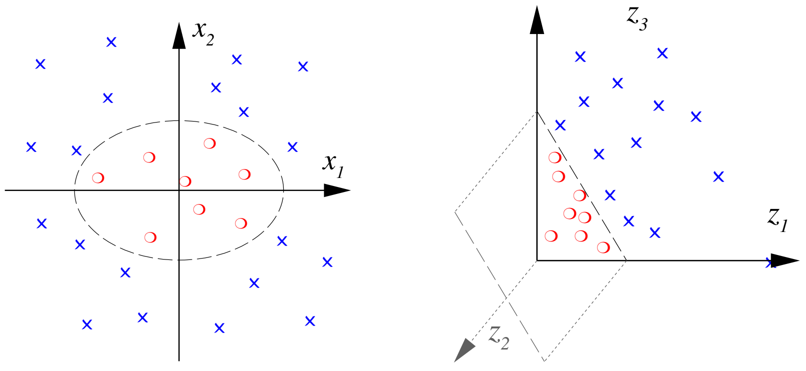
How about patterns $x \in \mathbb{R}^N$ and product features of order d?

Here, $\dim(\mathcal{H})$ grows like N^d.

E.g. $N = 16 \times 16$, and $d = 5 \longrightarrow$ dimension 10^{10}

Example: All Degree 2 Monomials

$$\begin{aligned} \Phi : \mathbb{R}^2 &\to \mathbb{R}^3 \\ (x_1, x_2) &\mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2) \end{aligned}$$



The Kernel Trick, $N = d = 2$

$$\begin{aligned} \left\langle \Phi(x), \Phi(x') \right\rangle &= (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)(x_1'^2, \sqrt{2}\, x_1' x_2', x_2'^2)^\top \\ &= \left\langle x, x' \right\rangle^2 \\ &=: k(x, x') \end{aligned}$$

$\longrightarrow$ the dot product in $\mathcal{H}$ can be computed in $\mathbb{R}^2$

The Kernel Trick, II

More generally: $x, x' \in \mathbb{R}^N$, $d \in \mathbb{N}$:

$$\langle x, x' \rangle^d = \left(\sum_{j=1}^{N} x_j \cdot x'_j \right)^d$$
$$= \sum_{j_1, \ldots, j_d = 1}^{N} x_{j_1} \cdot \cdots \cdot x_{j_d} \cdot x'_{j_1} \cdot \cdots \cdot x'_{j_d} = \langle \Phi(x), \Phi(x') \rangle ,$$

where Φ maps into the space spanned by all ordered products of d input directions

The Mercer Feature Map

In that case

$$\Phi(x) := \begin{pmatrix} \sqrt{\lambda_1}\psi_1(x) \\ \sqrt{\lambda_2}\psi_2(x) \\ \vdots \end{pmatrix}$$

satisfies $\langle \Phi(x), \Phi(x') \rangle = k(x, x')$.

Proof:

$$\langle \Phi(x), \Phi(x') \rangle = \left\langle \begin{pmatrix} \sqrt{\lambda_1}\psi_1(x) \\ \sqrt{\lambda_2}\psi_2(x) \\ \vdots \end{pmatrix}, \begin{pmatrix} \sqrt{\lambda_1}\psi_1(x') \\ \sqrt{\lambda_2}\psi_2(x') \\ \vdots \end{pmatrix} \right\rangle$$
$$= \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i(x') = k(x, x')$$

Mercer's Theorem

If k is a continuous kernel of a positive definite integral operator on $L_2(\mathcal{X})$ (where $\mathcal{X}$ is some compact space),

$$\int_{\mathcal{X}} k(x, x') f(x) f(x') \, dx \, dx' \geq 0,$$

it can be expanded as

$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i(x')$$

using eigenfunctions ψ_i and eigenvalues $\lambda_i \geq 0$ [38].

The Kernel Trick — Summary

- *any* algorithm that only depends on dot products can benefit from the kernel trick
- this way, we can apply linear methods to vectorial as well as *non-vectorial data*
- think of the kernel as a nonlinear *similarity measure*
- examples of common kernels:

$$\begin{aligned} \text{Polynomial} \quad k(x, x') &= (\langle x, x' \rangle + c)^d \\ \text{Sigmoid} \quad k(x, x') &= \tanh(\kappa \langle x, x' \rangle + \Theta) \\ \text{Gaussian} \quad k(x, x') &= \exp(-\|x - x'\|^2 / (2\,\sigma^2)) \end{aligned}$$

- Kernels are studied also in approximation theory *[39, 66, 4]* and in the Gaussian Process prediction community (covariance functions) *[69, 66, 71, 37]* — cf. the courses of Carl Rasmussen and Alex Smola

Positive Definite Kernels

It can be shown that the class of kernels corresponding to a dot product in some feature space coincides with the one of positive definite (pd) kernels: kernels which are symmetric, and for

- any set of training points $x_1, \dots, x_m \in \mathcal{X}$ and
- any $a_1, \dots, a_m \in \mathbb{R}$

satisfy

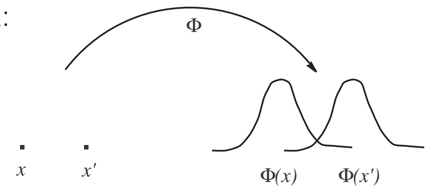
$$\sum_{i,j} a_i a_j K_{ij} \geq 0, \quad \text{where } K_{ij} := k(x_i, x_j).$$

The Feature Space for PD Kernels [4, 1, 46]

- define a feature map

$$\begin{aligned} \Phi : \mathcal{X} &\to \mathbb{R}^{\mathcal{X}} \\ x &\mapsto k(., x). \end{aligned}$$

E.g., for the Gaussian kernel:



Next steps:

- turn $\Phi(\mathcal{X})$ into a linear space
- endow it with a dot product satisfying $\langle k(., x_i), k(., x_j) \rangle = k(x_i, x_j)$
- complete the space to get a *reproducing kernel Hilbert space*

Elementary Properties of PD Kernels

Kernels from Feature Maps.
If Φ maps $\mathcal{X}$ into a dot product space $\mathcal{H}$, then $\langle \Phi(x), \Phi(x') \rangle$ is a pd kernel on $\mathcal{X} \times \mathcal{X}$.

Positivity on the Diagonal.
$k(x, x) \geq 0$ for all $x \in \mathcal{X}$

Cauchy-Schwarz Inequality.
$k(x, x')^2 \leq k(x, x)k(x', x')$ (Hint: compute the determinant of the Gram matrix)

Vanishing Diagonals.
$k(x, x) = 0$ for all $x \in \mathcal{X} \Longrightarrow k(x, x') = 0$ for all $x, x' \in \mathcal{X}$

Turn it Into a Linear Space

Form linear combinations

$$f(.) = \sum_{i=1}^{m} \alpha_i k(., x_i),$$

$$g(.) = \sum_{j=1}^{m'} \beta_j k(., x'_j)$$

$(m, m' \in \mathbb{N},\ \alpha_i, \beta_j \in \mathbb{R},\ x_i, x'_j \in \mathcal{X}).$

Endow it With a Dot Product

$$\langle f, g \rangle := \sum_{i=1}^{m} \sum_{j=1}^{m'} \alpha_i \beta_j k(x_i, x'_j)$$
$$= \sum_{i=1}^{m} \alpha_i g(x_i) = \sum_{j=1}^{m'} \beta_j f(x'_j)$$

- This is well-defined, symmetric, and bilinear.

Endow it With a Dot Product, II

- It can be shown that $\langle .,. \rangle$ is a p.d. kernel on the set of functions $\{f(.) = \sum_{i=1}^{m} \alpha_i k(., x_i) | \alpha_i \in \mathbb{R}, x_i \in \mathcal{X}\}$:

$$\sum_{ij} \gamma_i \gamma_j \langle f_i, f_j \rangle = \left\langle \sum_i \gamma_i f_i, \sum_j \gamma_j f_j \right\rangle =: \langle f, f \rangle$$
$$= \left\langle \sum_i \alpha_i k(., x_i), \sum_i \alpha_i k(., x_i) \right\rangle = \sum_{ij} \alpha_i \alpha_j k(x_i, x_j) \geq 0$$

- furthermore, it is *strictly* positive definite:

  $f(x)^2 = \langle f, k(., x) \rangle^2 \leq \langle f, f \rangle \langle k(., x), k(., x) \rangle = \langle f, f \rangle k(x, x)$

  hence $\langle f, f \rangle = 0$ implies $f = 0$.

- Complete the space in the corresponding norm to get a Hilbert space $\mathcal{H}_k$.

The Reproducing Kernel Property

Two special cases:

- Assume
$$f(.) = k(., x).$$
  In this case, we have
$$\langle k(., x), g \rangle = g(x).$$
- If moreover
$$g(.) = k(., x'),$$
  we have
$$\langle k(., x), k(., x') \rangle = k(x, x').$$

k is called a *reproducing kernel*

Explicit Construction of the RKHS Map for Mercer Kernels

Recall that the dot product has to satisfy
$$\langle k(x, .), k(x', .) \rangle = k(x, x').$$

For a Mercer kernel
$$k(x, x') = \sum_{j=1}^{N_F} \lambda_j \psi_j(x) \psi_j(x')$$

(with $\lambda_i > 0$ for all i, $N_F \in \mathbb{N} \cup \{\infty\}$, and $\langle \psi_i, \psi_j \rangle_{L_2(\mathcal{X})} = \delta_{ij}$), this can be achieved by choosing $\langle .,. \rangle$ such that
$$\langle \psi_i, \psi_j \rangle = \delta_{ij} / \lambda_i.$$

ctd.

To see this, compute

$$\begin{aligned}\langle k(x,.), k(x',.)\rangle &= \left\langle \sum_i \lambda_i \psi_i(x)\psi_i, \sum_j \lambda_j \psi_j(x')\psi_j \right\rangle \\ &= \sum_{i,j} \lambda_i\lambda_j \psi_i(x)\psi_j(x')\langle \psi_i, \psi_j\rangle \\ &= \sum_{i,j} \lambda_i\lambda_j \psi_i(x)\psi_j(x')\delta_{ij}/\lambda_i \\ &= \sum_i \lambda_i \psi_i(x)\psi_i(x') \\ &= k(x,x').\end{aligned}$$

The Empirical Kernel Map

Recall the feature map

$$\begin{aligned}\Phi : \mathcal{X} &\to \mathbb{R}^{\mathcal{X}} \\ x &\mapsto k(.,x).\end{aligned}$$

- each point is represented by its similarity to *all* other points
- how about representing it by its similarity to a *sample* of points?

Consider

$$\begin{aligned}\Phi_m : \mathcal{X} &\to \mathbb{R}^m \\ x &\mapsto k(.,x)|_{(x_1,\dots,x_m)} = (k(x_1,x),\dots,k(x_m,x))^\top\end{aligned}$$

Deriving the Kernel from the RKHS

An RKHS is a Hilbert space $\mathcal{H}$ of functions f where all *point evaluation functionals*

$$\begin{aligned}p_x : \mathcal{H} &\to \mathbb{R} \\ f &\mapsto p_x(f) = f(x)\end{aligned}$$

exist and are continuous.

Continuity means that whenever f and f' are close in $\mathcal{H}$, then $f(x)$ and $f'(x)$ are close in $\mathbb{R}$. This can be thought of as a topological prerequisite for generalization ability.

By Riesz' representation theorem, there exists an element of $\mathcal{H}$, call it r_x, such that

$$\langle r_x, f\rangle = f(x) \quad \text{for all } f,$$

in particular,

$$\langle r_x, r_{x'}\rangle = r_{x'}(x).$$

Define $k(x,x') := r_x(x') = r_{x'}(x)$.

(cf. Canu & Mary, 2002)

ctd.

- $\Phi_m(x_1),\dots,\Phi_m(x_m)$ contain *all* necessary information about $\Phi(x_1),\dots,\Phi(x_m)$
- the Gram matrix $G_{ij} := \left\langle \Phi_m(x_i), \Phi_m(x_j)\right\rangle$ satisfies $G = K^2$ where $K_{ij} = k(x_i,x_j)$
- modify Φ_m to

$$\begin{aligned}\Phi_m^w : \mathcal{X} &\to \mathbb{R}^m \\ x &\mapsto K^{-\frac{1}{2}}(k(x_1,x),\dots,k(x_m,x))^\top\end{aligned}$$

- this "whitened" map ("kernel PCA map") satifies

$$\left\langle \Phi_m^w(x_i), \Phi_m^w(x_j)\right\rangle = k(x_i,x_j)$$

for all $i,j = 1,\dots,m$.

Some Properties of Kernels [50]

If $k_1, k_2, \ldots$ are pd kernels, then so are

- αk_1, provided $\alpha \geq 0$
- $k_1 + k_2$
- $k_1 \cdot k_2$
- $k(x, x') := \lim_{n\to\infty} k_n(x, x')$, provided it exists
- $k(A, B) := \sum_{x\in A, x'\in B} k_1(x, x')$, where A, B are finite subsets of $\mathcal{X}$
  (using the feature map $\tilde{\Phi}(A) := \sum_{x\in A} \Phi(x)$)

Further operations to construct kernels from kernels: tensor products, direct sums, convolutions [26].

Properties of Kernel Matrices, I [47]

Suppose we are given distinct training patterns $x_1, \ldots, x_m$, and a positive definite $m \times m$ matrix K.

K can be diagonalized as $K = SDS^\top$, with an orthogonal matrix S and a diagonal matrix D with nonnegative entries. Then

$$K_{ij} = (SDS^\top)_{ij} = \langle S_i, DS_j \rangle = \left\langle \sqrt{D}S_i, \sqrt{D}S_j \right\rangle,$$

where the S_i are the rows of S.

We have thus constructed a map Φ into an m-dimensional feature space $\mathcal{H}$ such that

$$K_{ij} = \langle \Phi(x_i), \Phi(x_j) \rangle .$$

Properties, II: Functional Calculus [53]

- K symmetric $m \times m$ matrix with eigenvalues in $[\lambda_{\min}, \lambda_{\max}]$
- f a continuous function on $[\lambda_{\min}, \lambda_{\max}]$
- Then there is a symmetric matrix $f(K)$ with eigenvalues in $f([\lambda_{\min}, \lambda_{\max}])$.
- compute $f(K)$ via Taylor series, or eigenvalue decomposition of K: If $K = S^\top DS$ (D diagonal and S unitary), then $f(K) = S^\top f(D)S$, where $f(D)$ is defined elementwise on the diagonal
- can treat functions of symmetric matrices like functions on $\mathbb{R}$

$$\begin{aligned}
(\alpha f + g)(K) &= \alpha f(K) + g(K) \\
(fg)(K) &= f(K)g(K) = g(K)f(K) \\
\|f\|_{\infty,\sigma(K)} &= \|f(K)\| \\
\sigma(f(K)) &= f(\sigma(K))
\end{aligned}$$

(the C^*-algebra generated by K is isomorphic to the set of continuous functions on $\sigma(K)$)

Computing Distances in Feature Spaces

Clearly, if k is positive definite, then there exists a map Φ such that

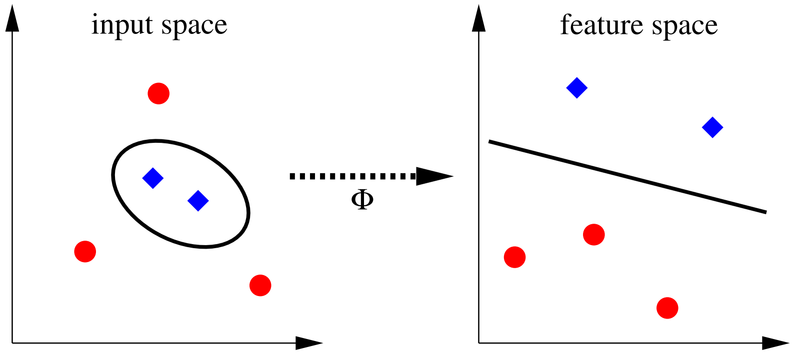
$$\|\Phi(x) - \Phi(x')\|^2 = k(x, x) + k(x', x') - 2k(x, x')$$

(it is the usual feature map).

This embedding is referred to as a *Hilbert space representation* as a distance. It turns out that this works for a larger class of kernels, called *conditionally positive definite*.

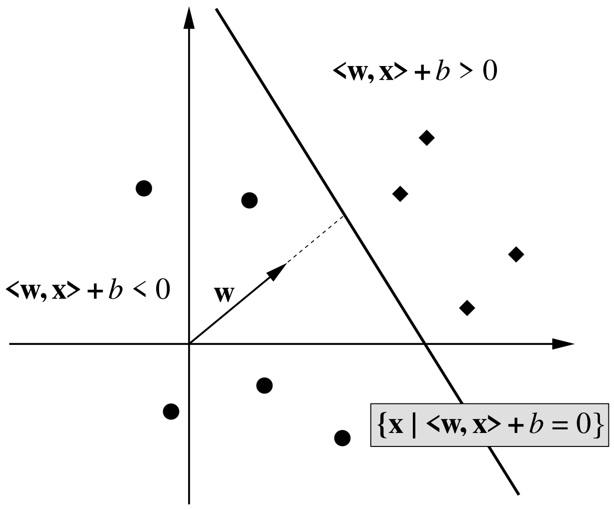
In fact, all algorithms that are translationally invariant (i.e. independent of the choice of the origin) in $\mathcal{H}$ work with cpd kernels [50].

Support Vector Classifiers

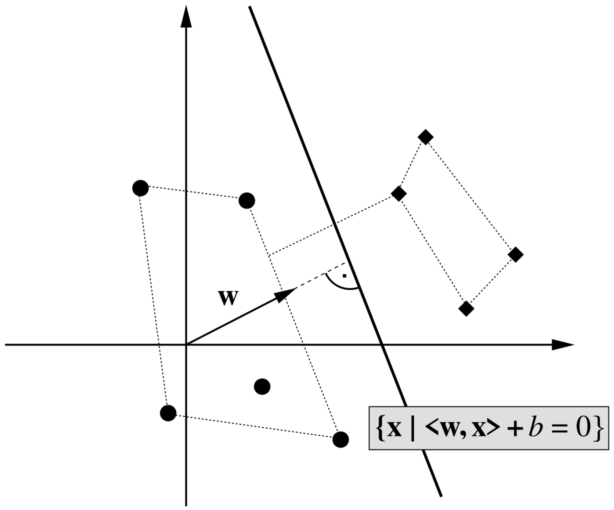


[6]

Separating Hyperplane



Optimal Separating Hyperplane



Eliminating the Scaling Freedom

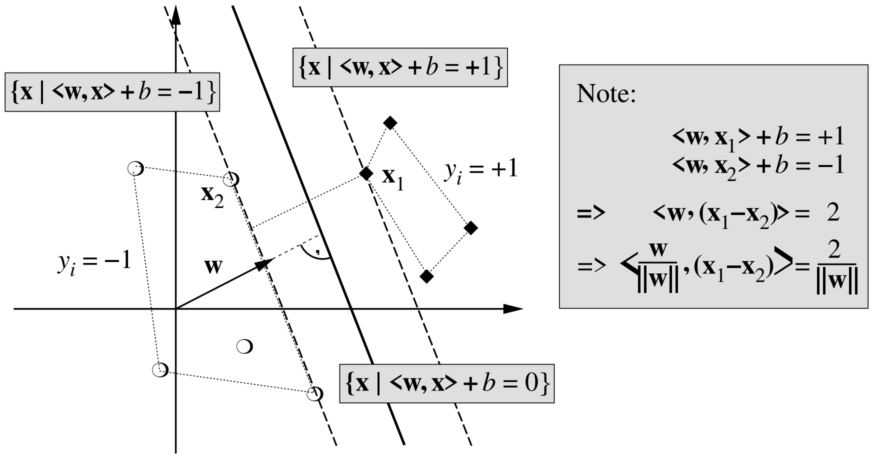
Note: if $c \neq 0$, then

$$\{\mathbf{x} | \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\} = \{\mathbf{x} | \langle c\mathbf{w}, \mathbf{x} \rangle + cb = 0\}.$$

Hence $(c\mathbf{w}, cb)$ describes the same hyperplane as $(\mathbf{w}, b)$.

Definition: The hyperplane is in *canonical* form w.r.t. $X^* = \{\mathbf{x}_1, \ldots, \mathbf{x}_r\}$ if $\min_{\mathbf{x}_i \in X} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b| = 1$.

Canonical Optimal Hyperplane



Lagrange Function (e.g., [5])

Introduce Lagrange multipliers $\alpha_i \geq 0$ and a Lagrangian

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i \left(y_i \cdot [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] - 1 \right).$$

L has to minimized w.r.t. the *primal variables* $\mathbf{w}$ and b and maximized with respect to the *dual variables* α_i

- if a constraint is violated, then $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 < 0 \longrightarrow$
  - $\cdot$ α_i will grow to increase L — how far?
  - $\cdot$ $\mathbf{w}$, b want to decrease L; i.e. they have to change such that the constraint is satisfied. If the problem is separable, this ensures that $\alpha_i < \infty$.
- similarly: if $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 > 0$, then $\alpha_i = 0$: otherwise, L could be increased by decreasing α_i *(KKT conditions)*

Formulation as an Optimization Problem

Hyperplane with maximum margin: minimize

$$\|\mathbf{w}\|^2$$

(recall: margin $\sim 1/\|\mathbf{w}\|$) subject to

$$y_i \cdot [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] \geq 1 \quad \text{for } i = 1 \ldots m$$

(i.e. the training data are separated correctly).

Derivation of the Dual Problem

At the extremum, we have

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \quad \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0,$$

i.e.

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

and

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i.$$

Substitute both into L to get the *dual problem*

The Support Vector Expansion

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

where for all $i = 1, \dots, m$ either

$y_i \cdot [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] > 1 \quad \Longrightarrow \alpha_i = 0 \longrightarrow \mathbf{x}_i$ irrelevant
or
$y_i \cdot [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] = 1$ (*on* the margin) $\longrightarrow \mathbf{x}_i$ "Support Vector"

The solution is determined by the examples on the margin.

Thus

$$\begin{aligned} f(\mathbf{x}) &= \mathrm{sgn}\left(\langle \mathbf{x}, \mathbf{w} \rangle + b\right) \\ &= \mathrm{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i \langle \mathbf{x}, \mathbf{x}_i \rangle + b\right). \end{aligned}$$

A Mechanical Interpretation

Assume that each SV $\mathbf{x}_i$ exerts a perpendicular force of size α_i and sign y_i on a solid plane sheet lying along the hyperplane.

Then the solution is mechanically stable:

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \quad \text{implies that the forces sum to zero}$$

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \quad \text{implies that the torques sum to zero,}$$

via

$$\sum_i \mathbf{x}_i \times y_i \alpha_i \cdot \mathbf{w}/\|\mathbf{w}\| = \mathbf{w} \times \mathbf{w}/\|\mathbf{w}\| = 0.$$

Why it is Good to Have Few SVs

Leave out an example that does not become SV $\longrightarrow$ same solution.

Theorem [60]: Denote $\#\mathrm{SV}(m)$ the number of SVs obtained by training on m examples randomly drawn from $\mathrm{P}(\mathbf{x}, y)$, and $\mathbf{E}$ the expectation. Then

$$\mathbf{E}\left[\mathrm{Prob}(\text{test error})\right] \le \frac{E\left[\#\mathrm{SV}(m)\right]}{m}$$

Here, Prob(test error) refers to the expected value of the risk, where the expectation is taken over training the SVM on samples of size $m - 1$.

Dual Problem

Dual: maximize

$$W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle$$

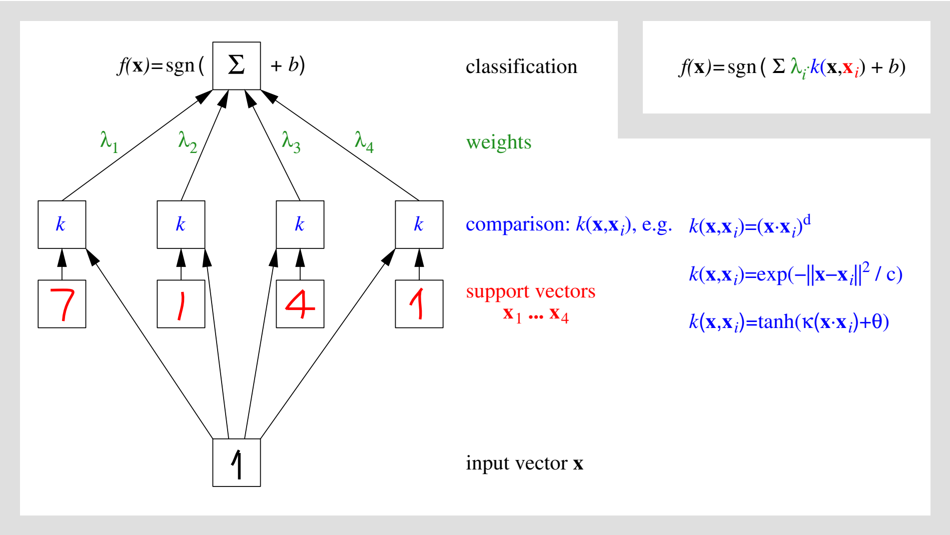
subject to

$$\alpha_i \ge 0, \quad i = 1, \dots, m, \quad \text{and} \quad \sum_{i=1}^{m} \alpha_i y_i = 0.$$

Both the final decision function and the function to be maximized are expressed in dot products $\longrightarrow$ can use a kernel to compute

$$\left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle = \left\langle \Phi(x_i), \Phi(x_j) \right\rangle = k(x_i, x_j).$$

The SVM Architecture



Nonseparable Problems [3, 16]

If $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$ cannot be satisfied, then $\alpha_i \to \infty$.

Modify the constraint to

$$y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$$

with

$$\xi_i \geq 0$$

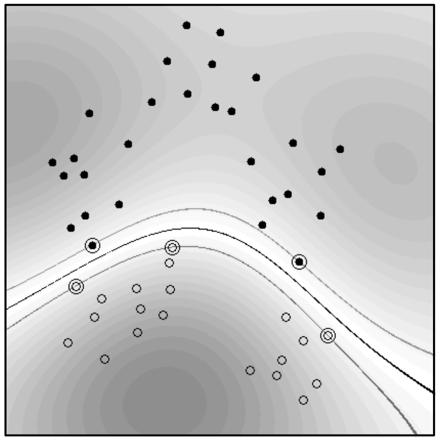
("*soft margin*") and add

$$C \cdot \sum_{i=1}^{m} \xi_i$$

in the objective function.

Toy Example with Gaussian Kernel

$$k(x, x') = \exp\left(-\|x - x'\|^2\right)$$



Soft Margin SVMs

C-SVM [16]: for $C > 0$, minimize

$$\tau(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \xi_i$$

subject to $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$ (margin $2/\|\mathbf{w}\|$)

ν-SVM [51]: for $0 \leq \nu < 1$, minimize

$$\tau(\mathbf{w}, \boldsymbol{\xi}, \rho) = \frac{1}{2}\|\mathbf{w}\|^2 - \nu\rho + \frac{1}{m} \sum_{i} \xi_i$$

subject to $y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \rho - \xi_i, \quad \xi_i \geq 0$ (margin $2\rho/\|\mathbf{w}\|$)

The ν-Property

SVs: $\alpha_i > 0$
"margin errors:" $\xi_i > 0$

KKT-Conditions $\Longrightarrow$

- All margin errors are SVs.
- Not all SVs need to be margin errors.
  Those which are *not* lie exactly on the edge of the margin.

Proposition:
1. *fraction of Margin Errors* $\leq \nu \leq$ *fraction of SVs.*
2. *asymptotically:* ... $= \nu =$ ...

Connection between ν-SVC and C-SVC

Proposition. If ν-SV classification leads to $\rho > 0$, then C-SV classification, with C set a priori to $1/\rho$, leads to the same decision function.

Proof. Minimize the primal target, then fix ρ, and minimize only over the remaining variables: nothing will change. Hence the obtained solution $\mathbf{w}_0, b_0, \boldsymbol{\xi}_0$ minimizes the primal problem of C-SVC, for $C = 1$, subject to

$$y_i \cdot (\langle \mathbf{x}_i, \mathbf{w} \rangle + b) \geq \rho - \xi_i.$$

To recover the constraint

$$y_i \cdot (\langle \mathbf{x}_i, \mathbf{w} \rangle + b) \geq 1 - \xi_i,$$

rescale to the set of variables $\mathbf{w}' = \mathbf{w}/\rho, b' = b/\rho, \boldsymbol{\xi}' = \boldsymbol{\xi}/\rho$. This leaves us, up to a constant scaling factor ρ^2, with the C-SV target with $C = 1/\rho$.

Duals, Using Kernels

C-SVM dual: maximize

$$W(\boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

subject to $0 \leq \alpha_i \leq C, \quad \sum_i \alpha_i y_i = 0.$

ν-SVM dual: maximize

$$W(\boldsymbol{\alpha}) = -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

subject to $0 \leq \alpha_i \leq \frac{1}{m}, \quad \sum_i \alpha_i y_i = 0, \quad \sum_i \alpha_i \geq \nu$

In both cases: *decision function*:

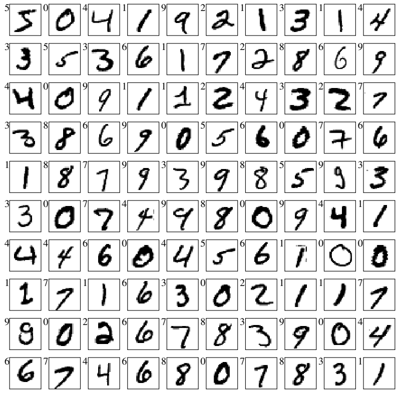
$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^m \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i) + b\right)$$

SVM Training

- naive approach: the complexity of maximizing
  $$W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$
  scales with the third power of the training set size m
- only SVs are relevant $\longrightarrow$ only compute $(k(\mathbf{x}_i, \mathbf{x}_j))_{ij}$ for SVs. Extract them iteratively by cycling through the training set in chunks [59].
- in fact, one can use chunks which do not even contain all SVs [40]. Maximize over these sub-problems, using your favorite optimizer.
- the extreme case: by making the sub-problems very small (just two points), one can solve them analytically [43].
- http://www.kernel-machines.org/software.html

MNIST Benchmark

handwritten character benchmark (60000 training & 10000 test examples, 28×28)



Speeding up the decision rule

Approximate
$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \Phi(x_i)$$
by
$$\mathbf{w}' = \sum_{i=1}^{N_z} \gamma_i \Phi(z_i),$$
with $N_z \ll m$: Minimize
$$\rho = \|\mathbf{w} - \mathbf{w}'\|^2$$
Note that ρ can be expressed in terms of k by using $\langle \Phi(x), \Phi(x') \rangle = k(x, x')$

Construct the new expansion sequentially.

"reduced set methods", [e.g. 10, 12, 41, 48]

MNIST Error Rates

Classifier	test error	reference
linear classifier	8.4%	[7]
3-nearest-neighbour	2.4%	[7]
SVM	1.4%	[12]
Tangent distance	1.1%	[55]
LeNet4	1.1%	[35]
Boosted LeNet4	0.7%	[35]
Translation invariant SVM	0.56%	[20]

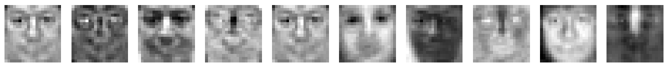
Note: the SVM used a polynomial kernel of degree 9, corresponding to a feature space of dimension $\approx 3.2 \cdot 10^{20}$.

Other successful applications: e.g., [31, 29, 27, 13, 57, 9, 74, 24, 23, 15, 22, 42, 67, 73]

Face Detection

- scan test images in several resolutions
- critical issue: runtime speed. Compute sequential approximation via reduced set expansion.
- need to evaluate on average 2 – 3 kernels per image location [45]

after 0, 1 *(13.3% patches remaining)*, 10 *(2.6%)*, 20 *(0.01%)* and 30 *(0.002%)* kernels

templates: 

Invariant Hyperplanes

Consider decision functions $f(x) = \mathrm{sgn}(g(x))$, where

$$g(x) := \sum_{i=1}^{m} v_i \, \langle Bx, Bx_i \rangle + b.$$

To get local invariance under transformations of the Lie group $\{\mathcal{L}_t\}$, minimize the regularizer

$$\frac{1}{m} \sum_{j=1}^{m} \left(\frac{\partial}{\partial t}\Big|_{t=0} g(\mathcal{L}_t x_j) \right)^2 .$$

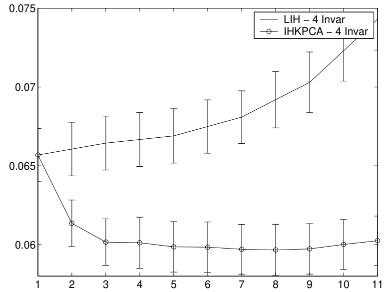
This corresponds to an SV optimization after preprocessing with

$$B = C^{-\frac{1}{2}},$$

where

$$C = \frac{1}{m} \sum_{j=1}^{m} \left(\frac{\partial}{\partial t}\Big|_{t=0} \mathcal{L}_t x_j \right) \left(\frac{\partial}{\partial t}\Big|_{t=0} \mathcal{L}_t x_j \right)^{\top} .$$

USPS Digit Recognition Application [14]



Results for 4 invariance transformations (translations) and different trade-offs between margin maximization and invariance enforcement (left: standard SVM).

The Tangent Covariance Matrix

C = covariance matrix of $\pm\frac{\partial}{\partial t}|_{t=0}\mathcal{L}_t x$

Preprocessing of x:

$$Bx = C^{-\frac{1}{2}} x = S D^{-\frac{1}{2}} S^{\top} x$$

1. project x onto the Eigenvectors of C
2. divide by the square roots of the Eigenvalues, i.e.: the directions of main variance of $\pm\frac{\partial}{\partial t}|_{t=0}\mathcal{L}_t x$ are scaled back

- in practice, use $C_\lambda := (1-\lambda)C + \lambda I$
- for the nonlinear case, use the kernel PCA map [14]

SV Regression: ε-Insensitive Loss *(Vapnik, 1995)*

Goal: generalize SV pattern recognition to regression, preserving the following properties:

- formulate the algorithm for the linear case, and then use kernel trick
- sparse representation of the solution in terms of SVs

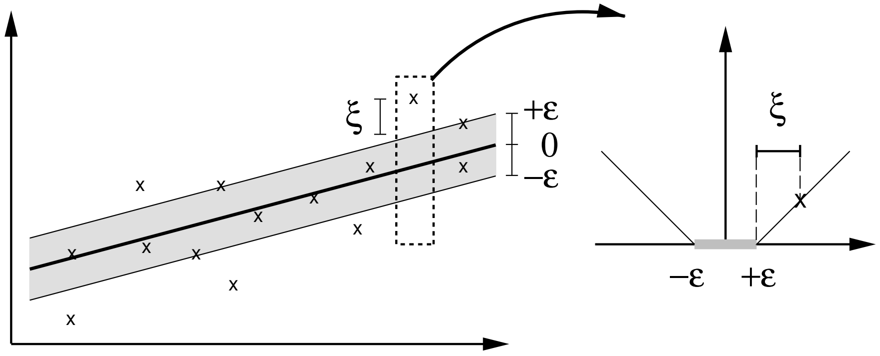
ε-Insensitive Loss:

$$|y - f(\mathbf{x})|_\varepsilon := \max\{0, |y - f(\mathbf{x})| - \varepsilon\}$$

Estimate a linear regression $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$ by minimizing

$$\frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m} \sum_{i=1}^{m} |y_i - f(\mathbf{x}_i)|_\varepsilon .$$

ε-SV Regression Estimation



Dual Problem, In Terms of Kernels

For $C > 0, \varepsilon \geq 0$ chosen a priori,

$$\text{maximize} \quad W(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = -\varepsilon \sum_{i=1}^{m} (\alpha_i^* + \alpha_i) + \sum_{i=1}^{m} (\alpha_i^* - \alpha_i) y_i - \frac{1}{2} \sum_{i,j=1}^{m} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) k(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to} \quad 0 \leq \alpha_i, \alpha_i^* \leq C, \; i = 1, \dots, m, \text{ and } \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) = 0.$$

The regression estimate takes the form

$$f(\mathbf{x}) = \sum_{i=1}^{m} (\alpha_i^* - \alpha_i) k(\mathbf{x}_i, \mathbf{x}) + b,$$

Formulation as an Optimization Problem

Estimate a linear regression

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

with precision ε by minimizing

$$\text{minimize} \quad \tau(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\xi}^*) = \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{m} (\xi_i + \xi_i^*)$$

$$\begin{aligned} \text{subject to} \quad & (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - y_i \leq \varepsilon + \xi_i \\ & y_i - (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \leq \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0 \end{aligned}$$

for all $i = 1, \dots, m$.

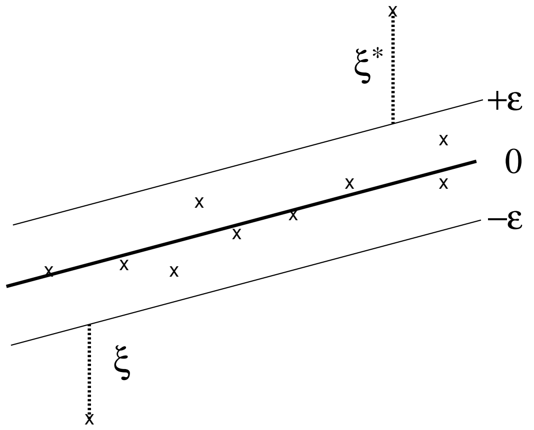
ν-SV Regression

Again, use ν to eliminate another parameter:
Estimate ε from the data s.t. the ν-property holds.

Primal problem: for $0 \leq \nu \leq 1$, minimize

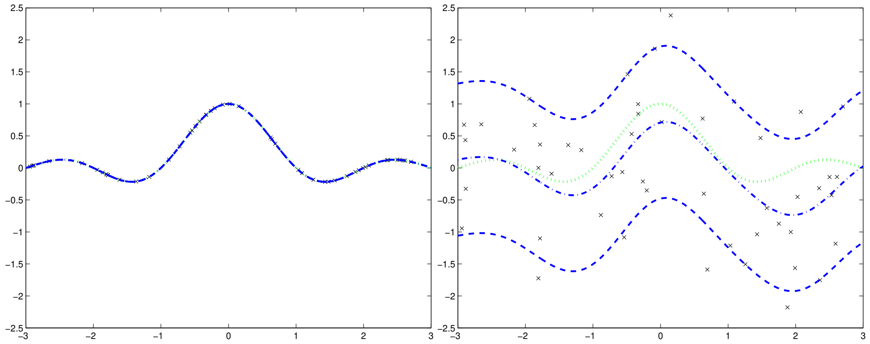
$$\tau(\mathbf{w}, \varepsilon) = \frac{1}{2}\|\mathbf{w}\|^2 + C \left(\nu\varepsilon + 1/m \sum_{i=1}^{m} |y_i - f(\mathbf{x}_i)|_\varepsilon \right)$$

A Graphical Proof of the ν-Property



Cost function: $\frac{1}{2C}\|\mathbf{w}\|^2 + \nu\varepsilon + \frac{1}{m}\sum_{i=1}^{m}(\xi_i + \xi_i^*)$

ν-SV-Regression: Automatic Tube Tuning



Identical machine parameters ($\nu = 0.2$), but different amounts of noise in the data.

The ν-Property

Proposition 3 *Assume $\varepsilon > 0$. The following statements hold:*

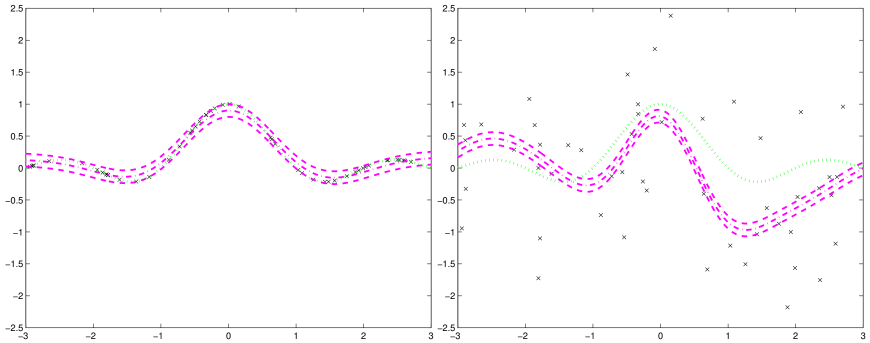
(i) ν is an upper bound on the fraction of errors.

(ii) ν is a lower bound on the fraction of SVs.

(iii) Suppose the data were generated iid from a 'well-behaved' distribution $\mathrm{P}(\mathbf{x}, y)$. With probability 1, asymptotically, ν equals both the fraction of SVs and the fraction of errors.*

* Essentially, $\mathrm{P}(\mathbf{x}, y) = \mathrm{P}(\mathbf{x})\mathrm{P}(y|\mathbf{x})$ with $\mathrm{P}(y|\mathbf{x})$ continuous (some details omitted).

ε-SV-Regression, Run on the Same Data



Identical machine parameters ($\varepsilon = 0.2$), but different amounts of noise in the data.

Toy Examples: Estimating a Noisy Sinc Function

$\nu = 0.2$

m	10	50	100	200	500	1000	1500	2000
ε	0.27	0.22	0.23	0.25	0.26	0.26	0.26	0.26
fraction of errors	0.00	0.10	0.14	0.18	0.19	0.20	0.20	0.20
fraction of SVs	0.40	0.28	0.24	0.23	0.21	0.21	0.20	0.20

- automatically computed ε largely independent of m
- asymptotics consistent with theorem

Comparison: ν vs. ε

ν-SVR	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
automatic ε	2.6	1.7	1.2	0.8	0.6	0.3	0.0	0.0	0.0	0.0
MSE	9.4	8.7	9.3	9.5	10.0	10.6	11.3	11.3	11.3	11.3
Errors	0.0	0.1	0.2	0.2	0.3	0.4	0.5	0.5	0.5	0.5
SVs	0.3	0.4	0.6	0.7	0.8	0.9	1.0	1.0	1.0	1.0

ε-SVR	0	1	2	3	4	5	6	7	8	9	10
MSE	11.3	9.5	8.8	9.7	11.2	13.1	15.6	18.2	22.1	27.0	34.3
Errors	0.5	0.2	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SVs	1.0	0.6	0.4	0.3	0.2	0.1	0.1	0.1	0.1	0.1	0.1

- RBF kernel, C and σ chosen as in [52]

Boston Housing Benchmark

- 506 examples, 13-dimensional.

Results (MSE):

- Bagging regression trees: 11.7 [8]
- ε-SV regression: 7.6 [56]

- 100 runs, with 25 randomly selected test points.
- training set is split into actual training set and validation set (80 points) for selecting ε, C, and kernel parameters
- ftp://ftp.ics.uci.com/pub/machine-learning-databases/housing

Parametric Error Models

Use a tube of varying radius $\zeta(\mathbf{x}) \geq 0$:
minimize

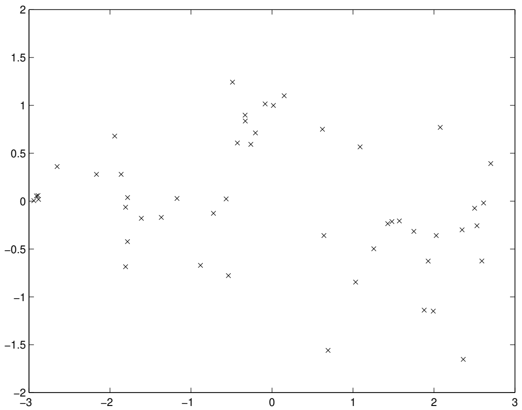
$$\tau(\mathbf{w}, \boldsymbol{\xi}^{(*)}, \varepsilon) = \|\mathbf{w}\|^2/2 + C \cdot \left(\nu m \varepsilon + \sum_{i=1}^{m} (\xi_i + \xi_i^*) \right)$$

$$\text{subject to} \quad \begin{aligned} (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - y_i &\leq \varepsilon \zeta(\mathbf{x}_i) + \xi_i \\ y_i - (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) &\leq \varepsilon \zeta(\mathbf{x}_i) + \xi_i^* \\ \xi_i^{(*)} &\geq 0, \quad \varepsilon \geq 0. \end{aligned}$$

This leads to the "usual" dual, with the modified last constraint

$$\sum_{i=1}^{m} (\alpha_i + \alpha_i^*) \zeta(\mathbf{x}_i) \leq C m \nu.$$

Toy Example: Some Noisy Data



Assumption: we have prior knowledge indicating that the noise is modulated by $\zeta(x) = \sin^2((2\pi/3)x)$.

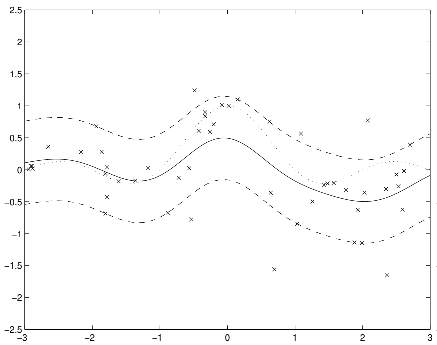
Robustness of SV Regression

Proposition. Using SVR with $|.|_\varepsilon$, local movements of target values of points outside the tube do not change the estimated regression.

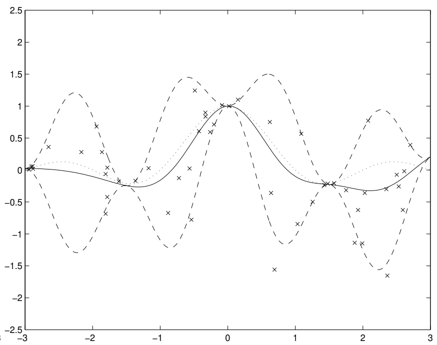
Proof.

1. Shift y_i locally $\longrightarrow$ $(\mathbf{x}_i, y_i)$ still outside the tube $\longrightarrow$ original dual solution $\boldsymbol{\alpha}^{(*)}$ still feasible ($\alpha_i^{(*)} = C$, since *all* points outside the tube are at the upper bound).
2. The primal solution, with ξ_i transformed according to the movement, is also feasible.
3. The KKT conditions are still satisfied, as still $\alpha_i^{(*)} = C$. Thus [5, e.g.], $\boldsymbol{\alpha}^{(*)}$ is still the optimal solution.

Toy Example, II



constant-radius tube



parametric model using $\zeta(x)$

The Representer Theorem

Theorem 4 *Given: a p.d. kernel k on $\mathcal{X} \times \mathcal{X}$, a training set $(x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{X} \times \mathbb{R}$, a strictly monotonic increasing real-valued function Ω on $[0, \infty[$, and an arbitrary cost function $c : (\mathcal{X} \times \mathbb{R}^2)^m \to \mathbb{R} \cup \{\infty\}$*

Any $f \in \mathcal{H}$ minimizing the regularized risk functional

$$c\left((x_1, y_1, f(x_1)), \ldots, (x_m, y_m, f(x_m))\right) + \Omega\left(\|f\|\right) \qquad (3)$$

admits a representation of the form

$$f(.) = \sum_{i=1}^{m} \alpha_i k(x_i, .).$$

Remarks

- significance: many learning algorithms have solutions that can be expressed as expansions in terms of the training examples
- original form, with mean squared loss

  $$c((x_1, y_1, f(x_1)), \ldots, (x_m, y_m, f(x_m))) = \frac{1}{m}\sum_{i=1}^{m}(y_i - f(x_i))^2,$$

  and $\Omega(\|f\|) = \lambda\|f\|^2$ ($\lambda > 0$): [33]
- generalization to non-quadratic cost functions: [17]
- present form: [50]

Proof

Decompose $f \in \mathcal{H}$ into a part in the span of the $k(x_i, .)$ and an orthogonal one:

$$f = \sum_i \alpha_i k(x_i, .) + f_\perp,$$

where for all j

$$\langle f_\perp, k(x_j, .)\rangle = 0.$$

Application of f to an arbitrary training point x_j yields

$$\begin{aligned} f(x_j) &= \langle f, k(x_j, .)\rangle \\ &= \left\langle \sum_i \alpha_i k(x_i, .) + f_\perp, k(x_j, .) \right\rangle \\ &= \sum_i \alpha_i \langle k(x_i, .), k(x_j, .)\rangle, \end{aligned}$$

independent of $f_\perp$.

Proof: second part of (3)

Since $f_\perp$ is orthogonal to $\sum_i \alpha_i k(x_i, .)$, and Ω is strictly monotonic, we get

$$\begin{aligned} \Omega(\|f\|) &= \Omega\left(\|\sum_i \alpha_i k(x_i, .) + f_\perp\|\right) \\ &= \Omega\left(\sqrt{\|\sum_i \alpha_i k(x_i, .)\|^2 + \|f_\perp\|^2}\right) \\ &\geq \Omega\left(\|\sum_i \alpha_i k(x_i, .)\|\right), \end{aligned} \tag{4}$$

with equality occuring if and only if $f_\perp = 0$.
Hence, any minimizer must have $f_\perp = 0$. Consequently, any solution takes the form

$$f = \sum_i \alpha_i k(x_i, .).$$

Application 1: Support Vector Classification

Here, $y_i \in \{\pm 1\}$. Use

$$c\left((x_i, y_i, f(x_i))_i\right) = \frac{1}{\lambda}\sum_i \max\left(0, 1 - y_i f(x_i)\right),$$

and the regularizer $\Omega(\|f\|) = \|f\|^2$.
$\lambda \to 0$ leads to the hard margin SVM

Application 2: Bayesian MAP Estimates

Identify (3) with the negative log posterior (cf. Kimeldorf & Wahba, 1970, Poggio & Girosi, 1990), i.e.

- $\exp(-c((x_i, y_i, f(x_i))_i))$ — likelihood of the data
- $\exp(-\Omega(\|f\|))$ — prior over the set of functions
- $\Omega(\|f\|) = \lambda\|f\|^2$ — Gaussian process prior [71] with covariance function k
- $\Omega(\|f\|) = \lambda\|f\|$ — Laplacian prior
- minimizer of (3) = MAP estimate

cf. [32]

The SV Expansion in Feature Space

- due to the representer theorem, the solution of kernel algorithms usually corresponds to a single vector in $\mathcal{H}$

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \Phi(x_i).$$

However, there is usually no $x \in \mathcal{X}$ such that

$$\Phi(x) = \mathbf{w},$$

i.e., $\Phi(\mathcal{X})$ is not closed under linear combinations — it is a nonlinear manifold (cf. [11, 48]).

Application 3: Kernel PCA

Kernel PCA can be shown to correspond to the case of

$$c((x_i, y_i, f(x_i))_{i=1,\ldots,m}) = \begin{cases} 0 & \text{if } \frac{1}{m}\sum_i \left(f(x_i) - \frac{1}{m}\sum_j f(x_j)\right)^2 = 1 \\ \infty & \text{otherwise} \end{cases}$$

with g an arbitrary strictly monotonically increasing function.

Further Kernel Algorithms — Design Principles

1. "Kernel module"
   - similarity measure $k(x, x')$, where $x, x' \in \mathcal{X}$
   - data representation (in associated feature space where $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$) — thus can construct geometric algorithms
   - function class ("representer theorem," $f(x) = \sum_i \alpha_i k(x, x_i)$)
2. "Learning module"
   - classification
   - quantile estimation / novelty detection
   - feature extraction
   - ...

Kernel Dependency Estimation *[70]*

Given two sets $\mathcal{X}$ and $\mathcal{Y}$ with kernels k and k', and training data (x_i, y_i).

Estimate a dependency $\mathbf{w} : \mathcal{H} \to \mathcal{H}'$

$$\mathbf{w}(\cdot) = \sum_{ij} \alpha_{ij} \Phi'(y_j) \langle \Phi(x_i), \cdot \rangle .$$

This can be evaluated in various ways, e.g., given an x, we can compute the pre-image

$$y = \mathrm{argmin}_{\mathcal{Y}} \|\mathbf{w}(\Phi(x)) - \Phi'(y)\|.$$

A convenient way of learning the α_{ij} is to work in the kernel PCA basis.

Kernel Machines Research

- algorithms/tasks: KDE, feature selection *(Weston et al., 2002)*, multi-label-problems *(Elisseeff & Weston, 2001)*, unlabelled data *(Szummer & Jaakkola, 2002)*, ICA *[25]*, canonical correlations *(Bach & Jordan, 2002; Kuss, 2002)*
- optimization and implementation: QP, SDP *(Lanckriet et al., 2002)*, online versions, ...
- theory of empirical inference: sharper capacity measures and bounds *(Bartlett, Bousquet, & Mendelson, 2002)*, generalized evaluation spaces *(Mary & Canu, 2002)*, ...
- kernel design
  - transformation invariances *[14]*
  - kernels for discrete objects *[26, 68, 36, 19, 64]*
  - kernels based on generative models *[30, 54, 58]*
  - local kernels *[e.g., 74]*
  - complex kernels from simple ones *[26, 2]*, global kernels from local ones *[34]*
  - functional calculus for kernel matrices *[53]*
  - model selection, e.g., via alignment *[18]*

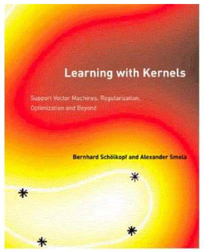
Application to Image Completion

Shown are all digits where at least one of the two algorithms makes a mistake (73 mistakes for k-NN, 23 for KDE).

(from [70])

Conclusion

- crucial ingredients of SV algorithms: kernels that can be represented as dot products, and large margin regularizers
- large margin regularizers ensure state of the art generalization ability even for high-dimensional data
- kernels allow the formulation of a multitude of geometrical algorithms (Parzen windows, SV pattern recognition, SV quantile estimation, kernel PCA,...) on arbitrary data types



For further information, cf.
`http://www.kernel-machines.org`,
`http://www.learning-with-kernels.org`

References

[1] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

[2] P. L. Bartlett and B. Schölkopf. Some kernels for structured data. Technical report, Biowulf Technologies, 2001.

[3] K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[4] C. Berg, J. P. R. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups*. Springer-Verlag, New York, 1984.

[5] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.

[6] B. E. Boser, I. M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, July 1992. ACM Press.

[7] L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. LeCun, U. A. Müller, E. Säckinger, P. Simard, and V. Vapnik. Comparison of classifier methods: a case study in handwritten digit recognition. In *Proceedings of the 12th International Conference on Pattern Recognition and Neural Networks, Jerusalem*, pages 77–87. IEEE Computer Society Press, 1994.

[8] L. Breiman. Bagging predictors. Technical Report 421, Department of Statistics, UC Berkeley, 1994. ftp://ftp.stat.berkeley.edu/pub/tech-reports/421.ps.Z.

[9] M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. S. Furey, M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data using support vector machines. *Proceedings of the National Academy of Sciences*, 97(1):262–267, 2000.

[10] C. J. C. Burges. Simplified support vector decision rules. In L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning*, pages 71–77, San Mateo, CA, 1996. Morgan Kaufmann.

[11] C. J. C. Burges. Geometry and invariance in kernel based methods. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 89–116, Cambridge, MA, 1999. MIT Press.

[12] C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector learning machines. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 375–381, Cambridge, MA, 1997. MIT Press.

[13] O. Chapelle, P. Haffner, and V. Vapnik. SVMs for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5), 1999.

[14] O. Chapelle and B. Schölkopf. Incorporating invariances in nonlinear SVMs. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002. To appear.

[15] S. Chen and C. J. Harris. Design of the optimal separating hyperplane for the decision feedback equalizer using support vector machines. In *IEEE International Conference on Acoustic, Speech, and Signal Processing*, Istanbul, Turkey, 2000.

[16] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[17] D. Cox and F. O'Sullivan. Asymptotic analysis of penalized likelihood and related estimators. *Annals of Statistics*, 18:1676–1695, 1990.

[18] N. Cristianini, A. Elisseeff, and J. Shawe-Taylor. On optimizing kernel alignment. Technical Report 2001-087, NeuroCOLT, 2001.

[19] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, Cambridge, UK, 2000.

[20] D. DeCoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46:161–190, 2002. Also: Technical Report JPL-MLTR-00-1, Jet Propulsion Laboratory, Pasadena, CA, 2000.

[21] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*, volume 31 of *Applications of mathematics*. Springer, New York, 1996.

[22] H. Drucker, B. Shahrary, and D. C. Gibbon. Relevance feedback using support vector machines. In *Proceedings of the 18th International Conference on Machine Learning*. Morgan Kaufmann, 2001.

[23] T. S. Furey, N. Duffy, N. Cristianini, D. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.

[24] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

[25] S. Harmeling, A. Ziehe, M. Kawanabe, and K.-R. Müller. Kernel feature spaces and nonlinear blind source separation. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002. To appear.

[26] D. Haussler. Convolutional kernels on discrete structures. Technical Report UCSC-CRL-99-10, Computer Science Department, University of California at Santa Cruz, 1999.

[27] M. A. Hearst, B. Schölkopf, S. Dumais, E. Osuna, and J. Platt. Trends and controversies — support vector machines. *IEEE Intelligent Systems*, 13:18–28, 1998.

[28] I. A. Ibragimov and R. Z. Has'minskii. *Statistical Estimation — Asymptotic Theory*. Springer-Verlag, New York, 1981.

[29] T. S. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7:95–114, 2000.

[30] T. S. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *Proceedings of the 1999 Conference on AI and Statistics*, 1999.

[31] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of the European Conference on Machine Learning*, pages 137–142, Berlin, 1998. Springer.

[32] G. S. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *Annals of Mathematical Statistics*, 41:495–502, 1970.

[33] G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33:82–95, 1971.

[34] I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of ICML'2002*, 2002.

[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.

[36] H. Lodhi, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. Technical Report 2000-79, NeuroCOLT, 2000. Published in: T. K. Leen, T. G. Dietterich and V. Tresp (eds.), *Advances in Neural Information Processing Systems 13*, MIT Press, 2001.

[37] D. J. C. MacKay. Introduction to Gaussian processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, pages 133–165. Springer-Verlag, Berlin, 1998.

[38] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, A 209:415–446, 1909.

[39] C. A. Micchelli. Algebraic aspects of interpolation. *Proceedings of Symposia in Applied Mathematics*, 36:81–102, 1986.

[40] E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. Technical Report AIM-1602, MIT A.I. Lab., 1996.

[41] E. Osuna and F. Girosi. Reducing run-time complexity in SVMs. In *Proceedings of the 14th Int'l Conf. on Pattern Recognition, Brisbane, Australia*, 1998.

[42] P. Pavlidis, J. Weston, J. Cai, and W. N. Grundy. Gene functional classification from heterogeneous data. In *Proceedings of the Fifth International Conference on Computational Molecular Biology*, pages 242–248, 2001.

[43] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.

[44] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9), September 1990.

[45] S. Romdhani, B. Schölkopf, P. Torr, and A. Blake. Fast face detection, using a sequential reduced support vector evaluation. TR 73, Microsoft Research, Redmond, WA, 2000. Published as: Computationally efficient face detection, *Proceedings of the International Conference on Computer Vision 2001*, pp. 695–700.

[46] S. Saitoh. *Theory of Reproducing Kernels and its Applications*. Longman Scientific & Technical, Harlow, England, 1988.

[47] B. Schölkopf. *Support Vector Learning*. R. Oldenbourg Verlag, München, 1997. Doktorarbeit, Technische Universität Berlin. Available from http://www.kyb.tuebingen.mpg.de/∼bs.

[48] B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999.

[49] B. Schölkopf, P. Simard, A. J. Smola, and V. Vapnik. Prior knowledge in support vector kernels. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 640–646, Cambridge, MA, 1998. MIT Press.

[50] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[51] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.

[52] B. Schölkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing*, 45:2758–2765, 1997.

[53] B. Schölkopf, J. Weston, E. Eskin, C. Leslie, and W. S. Noble. A kernel approach for learning from almost orthogonal patterns. In *Proceedings of the 13th European Conference on Machine Learning (ECML'2002) and Proceedings of the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'2002), Helsinki*, volume 2430/2431 of *Lecture Notes in Computer Science*, Berlin, 2002. Springer.

[54] M. Seeger. Bayesian methods for support vector machines and Gaussian processes. Master's thesis, University of Edinburgh, Division of Informatics, 1999.

[55] P. Simard, Y. LeCun, and J. Denker. Efficient pattern recognition using a new transformation distance. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5. Proceedings of the 1992 Conference*, pages 50–58, San Mateo, CA, 1993. Morgan Kaufmann.

[56] M. Stitson, A. Gammerman, V. Vapnik, V. Vovk, C. Watkins, and J. Weston. Support vector regression with ANOVA decomposition kernels. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 285–292, Cambridge, MA, 1999. MIT Press.

[57] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In P. Langley, editor, *Proceedings of the 17th International Conference on Machine Learning*, San Francisco, California, 2000. Morgan Kaufmann.

[58] K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K.R. Müller. A new discriminative kernel from probabilistic models. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002.

[59] V. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian]*. Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982).

[60] V. Vapnik and A. Chervonenkis. *Theory of Pattern Recognition [in Russian]*. Nauka, Moscow, 1974. (German Translation: W. Wapnik & A. Tscherwonenkis, *Theorie der Zeichenerkennung*, Akademie–Verlag, Berlin, 1979).

[61] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963.

[62] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

[63] V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[64] J.-P. Vert. A tree kernel to analyze phylogenetic profiles. In *Proceedings of ISMB'02*, 2002.

[65] U. von Luxburg, O. Bousquet, and B. Schölkopf. A compression approach to support vector model selection. Technical report, Max Planck Institute for Biological Cybernetics, 2002.

[66] G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1990.

[67] M. K. Warmuth, G. Rätsch, M. Mathieson, J. Liao, and C. Lemmen. Active learning in the drug discovery process. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002. To appear.

[68] C. Watkins. Dynamic alignment kernels. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 39–50, Cambridge, MA, 2000. MIT Press.

[69] H. L. Weinert, editor. *Reproducing Kernel Hilbert Spaces — Applications in Statistical Signal Processing*. Hutchinson Ross, Stroudsburg, PA, 1982.

[70] J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik. Kernel dependency estimation. Technical Report 98, Max Planck Institute for Biological Cybernetics, 2002.

[71] C. K. I. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning and Inference in Graphical Models*. Kluwer, 1998.

[72] D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996.

[73] C.-H. Yeang, S. Ramaswamy, P. Tamayo, S. Mukherjee, R. M. Rifkin, M. Angelo, M. Reich, E. Lander, J. Mesirov, and T. Golub. Molecular classification of multiple tumor types. *Bioinformatics*, 17:S316–S322, 2001. ISMB'01 Supplement.

[74] A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, 16(9):799–807, 2000.