T-61.5040

Learning Models and Methods, (5 cr) L

Spring 2007

Lecturer: **Petteri Pajunen**

Course Assistant: **Ville Viitaniemi**

LECTURE 5: 14.2.2007

STATISTICAL LEARNING THEORY
SUPPORT VECTOR MACHINE

Statistical Learning Theory

- motivation:
  - there are some "theories of learning" that seem to support learning from data without information
  - contradiction with NFL thms: important to examine what these theories actually claim
  - we concentrate on Statistical Learning Theory, perhaps the most well-known theory
  - Bayes-approach is used to explain the apparent conflict with No Free Lunch theorems

- Statistical Learning Theory contains results on generalizing a finite set of training data
- SLT $\overset{?}{\Longrightarrow}$ model-free learning is possible?
- not so: SLT does not make such a claim about learning from a *given set of data*
- SLT is not incorrect, but irrelevant to the learning problem defined earlier

- most details are skipped, as SLT turns out not to contradict NFL theorems
- the goal is to understand what the main result of SLT actually claims
- after this, you should be better equipped to decide whether studying such theories any further is worth the effort
- we can't skip SLT completely, since it is still being used to justify various model-free learning methods

- lets concentrate on binary outputs (binary regression or two-class classification)
- SLT requires that you predefine a set of solutions H which contains some (but not all) functions

$$h : X \to \{0, 1\}$$

- then SLT measures the "smallness" of the set H using Vapnik-Chervonenkis dimension

- SLT seems to say that if h fits training data well and is from a "small" set H, then h works well *in general*
- one wouldn't expect to find a good solution from a "small" H by chance
- note that this theory does not require you to know anything about the relationship between inputs $x \in X$ and the corresponding outputs in $\{0, 1\}$

Vapnik-Chervonenkis Dimension

- denote any training set of size n by

$$Z_n = \{(x_i, y_i) | x_i \in X,\ y_i \in \{0, 1\},\ i = 1, \ldots, n\}$$

- for each classifier $h \in H$ we obtain a *dichotomy*

$$(|y_1 - h(x_1)|, \ldots, |y_n - h(x_n)|)$$

- $|y_k - h(x_k)| = 1$ if h predicts wrong, otherwise zero
- fix Z_n, go through all $h \in H$ and denote the number of *different* dichotomies by $N(Z_n)$

- then maximize $N(Z_n)$ over *all sets* Z_n with n fixed
- we have gotten rid of the specific training set Z_n and the specific classifier h
- the *growth function*

$$G(n) = \log \max N(Z_n)$$

depends only on H, the domain X and the number of points n

- for any $n \geq 1$, one of the following holds:

$$G(n) = n \log 2$$
$$G(n) \leq v(\log(n/v) + 1)$$

- the integer v is the *VC-dimension*
- defined as the integer v satisfying

$$G(v) = v \log 2$$
$$G(v+1) < (v+1) \log 2$$

- example: H contains all linear classifiers on a plane
- any three points x_1, x_2, x_3 not on the same line can be classfied in all possible ways by classifiers from H
- then $G(3) = \log 2^3$
- any four points cannot be classified arbitrarily by lines
- therefore $G(4) < \log 2^4$ and the VC-dimension is three

- assume the "true" solution is defined by the unknown distribution $p(x, y)$
- the data $d = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ has been generated from the distribution $p(x, y)$
- define

$$c = \mathrm{E}(|y - h(x)|) \text{ (average error)}$$
$$s = \frac{1}{m} \sum_{i=1}^{m} |y_i - h(x_i)| \text{ (training error)}$$

- we want a small c, but can only compute s

- the quantity c is important, since it measures how well h performs *in general*, not just on the training set
- in our case of binary outputs, we have

$$\begin{aligned} c &= \mathrm{E}(|y - h(x)|) \\ &= \int \int |y - h(x)| p(x,y) dx dy \\ &= \mathbb{P}(|y - h(x)| = 1) \end{aligned}$$

- i.e. c is the *error rate* of the classifier h

- we oviously cannot compute c without knowing $p(x,y)$
- but we can compute s, the training error
- can we get information about c, given only training data d, set H, and training error s?
- NFL theorems say this is impossible

- main SLT result gives an upper bound for c as a function of s, m (number of training points), and the VC-dimension
- following inequality holds simultaneously for all $h \in H$ *with probability* $1 - \nu$:

$$c \leq s + \frac{\varepsilon}{2}\left[1 + \sqrt{1 + \frac{4s}{\varepsilon}}\right]$$

$$\varepsilon = 4\frac{G(2m) - \log(\nu/4)}{m}$$

- ε is a function of m, ν, and the VC-dimension

- example:

$$\begin{aligned} s &= 0, \text{ (no training error)} \\ v &= 1, \text{ (small VC-dimension)} \\ m &= 0.5 * \exp(10) \approx 11000 \text{ training points} \\ \nu/4 &= \exp(-4) \implies \nu \approx 0.07 \end{aligned}$$

- then the bound is

$$c \leq \varepsilon \leq 8e^{-10}(10 + 1 + 4) = 15 * 8 * e^{-10} \approx 0.005$$

- so perfect training performance, small VC-dimension and about eleven thousand samples seems to guarantee small c with probability $1 - \nu = 0.93$?

- is there a free lunch?
- it seems so: just select a set H with a small VC-dimension, then choose h which minimizes s
- the bound is often small, so we have found a good solution with no information?
- the set H can be freely chosen so this can't be explained by implicit information in selecting H

- unfortunately, there is no free lunch
- Bayes-approach: check what is known and what is uncertain
  - *known*: data d, set H, training error s, number of points m
  - *unknown*: average error c
- we should compute $p(c|d, H, s, m)$

- SLT uses only the *likelihood* $p(s|c, m, \ldots)$
- i.e. distribution of s as a function of c
- not surprising that $p(s|c, \ldots)$ is concentrated around c
- for example, $c = 0.1$ means the classifier makes an error once out of ten classifications on average
- not rocket science to expect that approximately one out of ten *training points* are misclassified
- so $p(s|c, \ldots)$ has a peak around c

- add the extra term to obtain $s + \frac{\epsilon}{2}[\ldots] > s$
- now $p(s + \frac{\epsilon}{2}[\ldots]|c, \ldots)$ is simply $p(s|c, \ldots)$ shifted to the right
- most of the probability mass is now on values larger than c
- so we can say $c \leq s + \frac{\epsilon}{2}[\ldots]$ with high probability

- *this is only true when s is random and depends on a constant c!*
- so we can say "if c is small, then s is probably small too"
- we *cannot* say "if s is small, then c is probably small"
- why? Conditional probabilities have to be reversed using Bayes' Theorem
- so we need to compute $p(c|s,\ldots)$ which requires $p(c)$

- example: four points, one classifier $h \in H = \{h\}$
- c is the number of errors divided by 4
- compute $p(s|c)$ on two training points:

c	0	0.25	0.5	0.75	1
$p(s=0\|c)$	1	0.5	1/6	0	0
$p(s=0.5\|c)$	0	0.5	2/3	0.5	0
$p(s=1\|c)$	0	0	1/6	0.5	1

- then $c \leq s + 1/4$ with probability at least 5/6

- suppose $s = 0$ so the bound is $c \leq 1/4$
- do we get $c \leq 1/4$ with probability at least $5/6$?

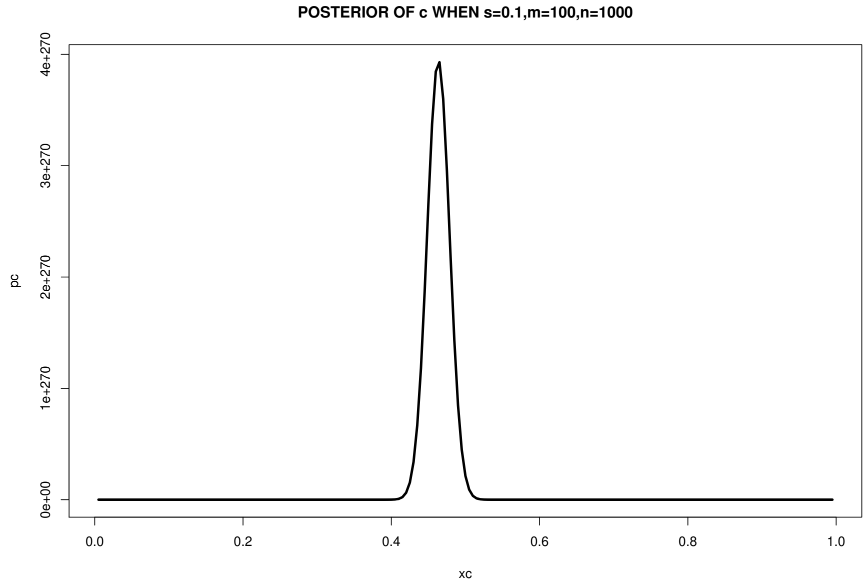
$$p(c = 0|s = 0) = 1/4$$
$$p(c = 1/4|s = 0) = 1/2$$

- since $3/4 < 5/6$, this demonstrates that we can't reverse probabilities without consequences

- the problem with SLT is similar to the fingerprint exercise problem
- $p(match|innocent)$ was small, but $p(innocent|match)$ was large
- reason was a very nonuniform prior of innocence
- in SLT, $p(c)$ is implicitly uniform, but it should not be

- uniform $p(c)$ implies that you have useful information about the error rate of h (exercise)
- if you don't, then a reasonable prior is a Binomial distribution for nc (exercise)
- this distribution is very nonuniform when n is even moderately large

- the flaw in SLT is demonstrated in detail in exercise problems
- example: binary classifier $H = \{h\}$
- $s = 0.1$, $m = 100$, and there are 1000 points to be classified
- small s and small VC-dimension v should imply that c is small?
- the posterior $p(c|d, s, m, v, h)$ looks like this (demo `fixed.R`):



- SLT bound gives $c \leq 0.33$ with probability $1 - \nu \approx 0.93$
- this clearly disagrees with the posterior
- the posterior would be concentrated *exactly* around $c = 0.5$, but c is measured *partly on the training set*
- *we already know* that for 100 training points the error is small ($s = 0.1$)

Support Vector Machine

- a two-class classifier often justified by SLT
- but we saw SLT does not show that learning without information is possible
- SVM has no properties that would give it a good off-training set error rate in general
- however, it has a number of practical advantages

- input vectors are mapped into a high-dimensional feature space using a *fixed nonlinear mapping*
- the fixed mapping is selected so that inner products in the feature space are easy to compute
- data is classified linearly in the feature space
- this corresponds to a nonlinear classifier in the data space

- SVM has some practical advantages:
  - flexible model: easy to implement complicated nonlinear classifiers
  - the classifier is typically determined by a small part of training data
  - computational benefits: quadratic programming solves the classifier, inner products are computed easily

Optimal Linear Classifier

- data: vectors $x_i \in \mathbb{R}^d$, class labels $y_i \in \{-1, 1\}$
- linearly separable training data $(x_1, y_1), \ldots, (x_m, y_m)$
- first component of x is always one: then a linear classifier is a hyperplane defined by $w'x = 0$
- the *normal vector* w defines the classifier

- x is classified by the sign of the inner product $w'x$
- OLC is defined as the linear classifier that
  1. makes no errors on the training set:

$$y_i w' x_i \geq 1, \quad i = 1, \ldots, m$$

  2. is as far as possible from the closest training point:

$$w^* = \text{argmax}_w \min_{i=1,\ldots,m} \frac{y_i w' x_i}{\|w\|}$$

- the expression $\frac{y_i w' x_i}{\|w\|}$ is the projection of x_i onto w, scaled by $\|w\|$
- this gives the distance of x_i to the hyperplane
- the above conditions result in the optimization problem

$$\frac{1}{2}\|w\|^2$$
$$y_i w' x_i \geq 1, \; i = 1, \ldots, m$$

- the solution is defined by the closest training point to the hyperplane
- there can be more than one: these points are called *support vectors*
- later we will see that the solution is in a sense defined by the support vectors

- OLC's "optimality": start with $S = \{w \mid y_i(w'x_i) \geq 1, i = 1, \ldots, m\}$
- VC-dimension of a subset $S(A) = \{w \in S \mid \|w\| \leq A\}$ gets smaller when A gets smaller
- SLT bound $c \leq s + \ldots$ is minimized when VC-dimension is minimized (due to $s = 0$)
- smallest $S(A) \neq \emptyset$ is $S(A = \min_{w \in S} \|w\|)$, so SLT suggests to minimize $\|w\|$ in S

- this derivation is useless: pick any $w_1 \in S$
- $S(A = \min_{w \in S} \|w\|)$ has at least the same VC-dimension as $\{w_1\}$
- so w_1 minimizes the bound as well as OLC
- all classifiers in S can't be "optimal" at the same time

- optimality of OLC fails in two ways:
  - having a small bound for c says nothing about the performance of the classifier (SLT's flaw)
  - even if it does, any classifier with $s = 0$ achieves the same bound
- OLC is optimal only in the sense it was defined before: it maximizes the distance to the closest training point

- regardless of optimality issues, we minimize

$$J = \frac{1}{2}\|w\|^2$$

with constraints

$$y_i(w'x_i) \geq 1,\ i = 1, \ldots, n$$

- constraints ensure $s = 0$ and minimizing J maximizes the distance to support vectors

- can be solved by Quadratic Programming, giving the solution in terms of Lagrange coefficients u_i^*:

$$w = \sum_i u_i^* y_i x_i$$

- most $u_i^* = 0$: the nonzero ones correspond to *support vectors* x_i
- new vectors can be classified by computing the sign of

$$w'x = \sum_i u_i^* y_i x_i' x$$

- to obtain SVM from this linear classifier, the vectors x_i are mapped nonlinearly to $z_i = g(x_i)$
- demo: `Kernels/polykern.R`
- inner products of z_i and z_j need to be computed:

$$H(x_i, x_j) = z_i' z_j = \sum_{k=1}^{D} g_k(x_i) g_k(x_j)$$

- using this directly is difficult if D is large (it often is)

- for some mappings $z = g(x)$, the *kernel* $H(x_i, x_j)$ is a simple function of x_i, x_j:
  - polynomials of degree q: $H = (1 + x_i' x_j)^q$
  - localized basis functions: $H(x, x_i) = \exp(-\|x - x_i\|^2 \sigma^{-2})$
  - Fourier-series: $H = \frac{\sin(q+0.5)(x_i - x_j)}{\sin(x_i - x_j)/2}$
- recall that we assumed that one component is always constant: this holds e.g. for the polynomial kernel since one basis function of a polynomial is a constant

- now SVM can be defined as follows:

1 choose a feature space for which there is an inner product kernel $H(\cdot, \cdot)$

2 compute an inner product matrix with components $H_{ij} = H(x_i, x_j)$

3 solve the quadratic optimization problem and obtain u_i^*'s

4 classify new vectors x by taking the sign of $\sum_i u_i^* y_i H(x_i, x)$

- benefits of SVM:
  - the dimension of z is irrelevant (computationally) when solving the optimization problem
  - the resulting classifier is nonlinear in x-space
  - new samples are easy to classify linearly using the kernel

- disadvantages:
  - nonlinear mapping g is chosen arbitrarily, but the solution depends on it
  - OLC is not optimal in a general sense: it is simply a linear classifier that maximizes the distance to closest training point
  - the maximum distance holds in the z-space, but in the x-space distances are different

- SVM demos: `Kernels/linear.R` (demos by Ralf Herbrich, somewhat modified)
- `exmp.svm1`: linear kernel
- `exmp.svm2`: polynomial kernel, $q = 7$
- `exmp.svm3`: RBF kernel