SRM and VC Theory (Statistical Learning Theory)

Robert C. Williamson*
Australian National University
Canberra, 0200 ACT, Australia

* Talk prepared whilst at Microsoft Research Cambridge
Aim

• Provide overview of statistical learning theory (VC and SRM Theory)
• Concentrate on the What and Why rather than the How
• Take an algorithmic point of view.
• Provoke Discussion
Outline

- Induction in General
- Problem of Learning and Risk Minimization
- ERM Inductive Principle
- SRM Inductive Principle
- Data-Dependent SRM
- Problems and Weaknesses
- Relationship with Other stuff
- Questions and Future Directions
Induction

Induction is the inference from empirical data. Generally impossible “for all is but a woven web of guesses”.

Popper’s key insight: scientific theories do not lead to certain knowledge; merely approximations to the truth.

We can, however, reason logically about the process of scientific discovery. Doing so shows one should prefer a more refutable theory over a less refutable one.

Based on work by Menger (1924) he even argued one could formally consider the “dimension” of a theory.

Although it is impossible to build a general learning machine Popper clearly admitted the possibility within a constrained framework of building a machine and being able to probabilistically reason about its performance.

We will study learning machines, and not induction in general.
The previous remarks are relevant so that one realises what statistical learning theory does not do: it does not provide a general theory of learning. It does provide considerable insight into the performance of various learning algorithms given that one is prepared to accept some logically unjustifiable assumptions to start with.

This can be viewed as a game. You can’t play the game without accepting the rules.

And you should never forget that you are playing a game.
The Formal Setup

Input space: \( \mathcal{X} \) often \( \mathbb{R}^n \).
Output space: \( \mathcal{Y} \) often \( \mathbb{R} \) or \( \{0, 1\} \).
Training data:

\[
\mathbf{z} := ((x_1, y_1), \ldots, x_m, y_m))
\]

where \((x_i, y_i) \in \mathcal{X} \times \mathcal{Y} =: \mathcal{Z}\) are generated iid according to a distribution \( \mathbb{P} \).
Aim: Given \( \mathbf{z} \), determine a function \( h \) such that

\[
R(h) := \int l(h(x), y) d\mathbb{P}(x, y)
\]

is small. \( R(h) \) is the \textit{expected risk} of \( h \) and \( l \) is the \textit{loss function}.
For example, \( l(h(x), y) = (h(x) - y)^2 \). We will assume \( l \) is always non-negative. We assume that \( \mathbb{P} \) can be arbitrary but is \textit{not} known.

The players . . . threw these abstract formulas at one another displaying the sequences and possibilities of their science.

— Herman Hesse: The Glass Bead Game
Risk Minimization

Each discipline which seized upon the Game created its own language of formulas, abbreviations, and possible combinations.

The (unrealizable) ideal is to use

$$A_{\text{ideal}} := z \mapsto \arg \min_{h \in Y^X} R(h)$$

Would even be happy if we had an algorithm $A$

$$P_z(R(A(z)) > \epsilon) \leq \delta$$

More generally have the “agnostic version”

$$P_z(R(A(z)) - R(h^*)) > \epsilon \leq \delta$$

where $h^*$ is some optimal function (optimal in the sense, for example, that it is best in some prespecified class $\mathcal{H}$: $h^* := \arg \min_{h \in \mathcal{H}} R(h)$).
How to Design Realizable $\mathcal{A}$?

Would like some guidance in designing $\mathcal{A}$. Unreasonable to expect to derive theoretically a priori a single algorithm with no user adjustable parameters.

In the spirit of good engineering, we would like a good user interface: we would like the parameters (knobs) to be “easy” to adjust.

Given $z \in \mathbb{Z}^m$ how can the algorithm $\mathcal{A}$ choose an hypothesis that achieves a small value of $R(\mathcal{A}(z))$? Key point: given $z \in \mathbb{Z}^m$. No chance of computing $R(\mathcal{A}(z))$ even in principle.
One Possible Solution...
 Attempted solution: Empirical Risk Minimization algorithm (ERM)

\[ \mathcal{A}_\text{erm}^\mathcal{H} : \mathcal{Z} \rightarrow \mathcal{H} \subseteq \mathcal{Y} \times \mathcal{X} \]

\[ \mathcal{A}_\text{erm}^\mathcal{H} : z \mapsto \arg \min_{h \in \mathcal{H}} R_{\text{emp}}(h, z) \]

where

\[ R_{\text{emp}}(h, z) = R_{\text{emp}}^m(h) = R_{\text{emp}}(h) := \frac{1}{m} \sum_{i=1}^{m} l(h(x_i), y_i) \]

and recall \( z = ((x_1, y_1), \ldots, (x_m, y_m)) \). Thus depending on the choice of sets \( \mathcal{H} \), get a family of empirical risk minimization algorithms.

Note that the algorithm \( \mathcal{A}_\text{erm}^\mathcal{H} \) has one “knob”: the class of functions \( \mathcal{H} \). We shall see below how to make a more readily graspable knob.

How to choose \( \mathcal{H} \)? Want \( \mathcal{H} \) as large as possible to ensure a good approximation of the underlying data generating process. Pay a price . . .
Consistency and Strict Consistency

Assume that for any $z \in \mathbb{Z}^m$, $A_{\text{erm}}^{\mathcal{H}}$ achieves the infimum in

$$\inf_{h \in \mathcal{H}} R_{\text{emp}}(h, z)$$

(reasonable!)

$$\mathcal{H}(c) := \{ h \in \mathcal{H} : R(h) \geq c \}$$

Say that $A_{\text{erm}}^{\mathcal{H}}$ is strictly (nontrivially) consistent if for all $c \geq 0$, for all $\epsilon > 0$,

$$\lim_{m \to \infty} \mathbb{P}_z \left( \left| R_{\text{emp}} \left( A_{\text{emp}}^{\mathcal{H}(c)}(z), z \right) - c \right| > \epsilon \right) = 0$$

Need a definition like this to rule out "coding" the identity of a function into one observation.

Can construct such artificial function classes of arbitrary complexity which can be learned using $A_{\text{erm}}^{\mathcal{H}}$ with only one observation.
Consistency of ERM

Assume that for all \( h \in \mathcal{H} \), \(|R(h)|\) is bounded.

\[ A_{\text{erm}}^{\mathcal{H}} \text{ is strictly consistent} \]

\[ \Longleftrightarrow \]

\[ \forall \epsilon > 0 \lim_{m \to \infty} \mathbb{P}_z \left\{ \sup_{h \in \mathcal{H}} (R(h) - R_{\text{emp}}^m(h, z)) > \epsilon \right\} = 0 \]

\[ \forall \epsilon > 0 \lim_{m \to \infty} \mathbb{P}_z \left\{ \sup_{h \in \mathcal{H}} |R(h) - R_{\text{emp}}^m(h, z)| > \epsilon \right\} = 0 \]

\[ \Longleftrightarrow \]

\[ \forall \epsilon > 0 \lim_{m \to \infty} \mathbb{E}_{z=(z_1, \ldots, z_m)} \log \mathcal{N}(\epsilon, \mathcal{H}, \ell_1(z)) = 0 \]

The effective gap in the reasoning implicit in the difference between \( \star \) and \( \star \star \) can be plugged using a more complex notion of cover — a one sided bracket cover. I am unaware of any results on the relative sizes of such covering numbers compared to \( \mathcal{N}(\epsilon, \mathcal{H}, \ell_1(z)) \).
So What?

The big deal is that (modulo the small gap mentioned)

$$\mathbb{E}_{\mathbf{z}=(z_1,\ldots,z_m)} \log N(\epsilon, \mathcal{H}, \ell_1(\mathbf{z}))$$

is the “right” quantity to study for understanding the effect of the $\mathcal{H}$ knob on $A_{\text{erm}}^\mathcal{H}$. (Why its worth fussing with strict consistency.)

Note that it is impossible to compute (even in principle) since we do not know $P$ (the distribution of $\mathbf{z}$).

Can upper bound by

$$\leq \sup_{\mathbf{z}=(z_1,\ldots,z_m)} \log N(\epsilon, \mathcal{H}, \ell_1(\mathbf{z})).$$

This can be bounded directly for (e.g. linear classes) or indirectly via the fat-shattering dimension

$$\leq c \text{fat}_\mathcal{H}(c\epsilon) \log^2 \left( \text{fat}_\mathcal{H}(c\epsilon)/\epsilon \right)$$
The “Key Theorem” in Learning Theory

“ERM is strictly consistent iff covering numbers behave nicely”

Observe that whilst we set out to understand the behaviour of \( A_{\text{erm}}^{\mathcal{H}} \), our bounds are in fact for \( \mathcal{A}_{\text{worst}}^{\mathcal{H}} := z \mapsto \arg \max_{h \in S(\mathcal{H}, z)} R(h) \)

where

\[
S(\mathcal{H}, z) = \{ h \in \mathcal{H} : R_{\text{emp}}(h, z) = R_{\text{emp}}(A_{\text{erm}}^{\mathcal{H}}, z) \}
\]

Consequently the bounds are very loose.

Furthermore \( A_{\text{erm}}^{\mathcal{H}} \) could perform as poorly as \( \mathcal{A}_{\text{worst}}^{\mathcal{H}} \) (what is there to stop it?).

Finally even for simple classes of functions (linear classifiers) \( A_{\text{erm}}^{\mathcal{H}} \) can be shown to be computationally complex.
Algorithm Independence of Bound

Since one exploits the key theorem to obtain upper bounds on $R(A_{\text{erm}}^H)$, could argue that the bounds are *algorithm independent* in the sense that the bound takes form: with probability $1 - \delta$ over a random draw of $z \in \mathcal{Z}^m$ according to $\mathbb{P}_z$, $R(A_{\text{erm}}^H(z)) \leq \psi(\mathcal{H}, z, \delta)$. Thus *any* algorithm

$$A_{\text{any}}^H : \mathcal{Z}^m \rightarrow \mathcal{H}$$

for which $R_{\text{emp}}(A_{\text{any}}^H(z), z) = 0$ has the same bound on performance. (Recall our earlier remark about ignoring the agnostic case in this talk.)

But classical bounds $\psi$ in terms of

$$\sup_{z=(z_1, \ldots, z_m)} \log \mathcal{N}(\epsilon, \mathcal{H}, \ell_1(z)) \quad \dagger$$

are justified via the key theorem — if you use $A_{\text{any}}^H$, then the key theorem does *not* tell you that $\dagger$ is the key knob to adjust.
Another Algorithm: SRM

An obvious difficulty with $A_{\text{erm}}^H$ is that if one chooses $H$ badly, the algorithm has no hope of approximating the data.

Suggests the following algorithm which uses $A_{\text{erm}}^H$ as a subroutine. Suppose for $z \in \mathcal{Z}^m$, we know

$$R\left(A_{\text{erm}}^H(z)\right) \leq \psi(H, z, \delta)$$

with probability $1 - \delta$ over a random draw of $z$ according to $P_z$.

Given a sequence of nonnegative numbers $\delta = (\delta_i)_{i \in \mathbb{N}}$ such that $\sum_i \delta_i = \delta$ and a sequence of hypothesis classes $\mathcal{H} = (\mathcal{H}_i)_{i \in \mathbb{N}}$

$$i^* = i^*(z, \mathcal{H}, \delta, \psi) := \arg\min_{i \in \mathbb{N}} \psi(\mathcal{H}_i, z, \delta_i)$$

$$A_{\text{erm}}^{H_{i^*}}(z) := A_{\text{erm}}^H(z)$$

SRM = Structural Risk Minimization
The good news about $A_{\text{srm}}^{{\mathcal{H}},\delta}$

By the definition of $A_{\text{srm}}^{{\mathcal{H}},\delta}$ it comes with a performance bound already. For $i \in \mathbb{N}$, with probability $1 - \delta_i$ over a random draw of $z$,

$$R \left( A_{\text{erm}}^{{\mathcal{H}_i}}(z) \right) \leq \psi(A_{\text{erm}}^{{\mathcal{H}_i}}, z, \delta_i)$$

Thus the union bound ensures that with probability $1 - \delta$ over a random draw of $z$, for all $i \in \mathbb{N}$

$$R \left( A_{\text{erm}}^{{\mathcal{H}_i}}(z) \right) \leq \psi(A_{\text{erm}}^{{\mathcal{H}_i}}, z, \delta_i)$$

and thus with probability $1 - \delta$ over a random draw of $z$,

$$R \left( A_{\text{srm}}^{{\mathcal{H}},\delta}(z) \right) \leq \psi(A_{\text{erm}}^{{\mathcal{H}_i*}}, z, \delta_{i*})$$

If one knows something about the approximation capabilities of $\mathcal{H}$ then can state some universal asymptotic consistency results.
The not so good news about $A_{srm}^{\mathcal{H},\delta}$

The strong justification of using bounds involving

$$\sup_{z=(z_1,\ldots,z_m)} \log \mathcal{N}(\epsilon, \mathcal{H}, \ell_1(z))$$

due to the key theorem in the case of $A_{erm}^{\mathcal{H}}$ no longer holds. Thus although we can get bounds like $\spadesuit$, we are not sure they are the best approach.

Note that the algorithm can not be used in a manner that ensures a bound on performance if $\mathcal{H}$ is allowed to depend on $z$.

Example: given $z \in \mathbb{Z}^m$, let $\mathcal{H}(z) = \{A_{erm}^{\mathcal{H}}(z)\}$ and let $\delta = (\delta, 0, 0, \ldots)$. Thus

$$A_{srm}^{\mathcal{H},\delta} = A_{erm}^{\mathcal{H}(z)}$$

and so $\dagger = 0$ leading to a crazy bound.

Need some constraint on how $\mathcal{H}$ can depend on $z$. 
Maximum Margin Algorithm

If there exists one separating hyperplane then there exist many of them. Why not choose the optimal one?
— Vladimir Vapnik: Statistical Learning Theory

Maximum Margin Classifier: $\mathcal{H}_i(z)$ comprises hyperplanes $h_w$ achieving margin $\gamma_i$ on $z$. Here $\gamma_z(h_w) := \max_{(x_i,y_i) \in z} y_i \langle w, \varphi(x_i) \rangle / \|w\|$. We know that

$$\psi(i) \leq \frac{c}{\gamma_i^2} \log^2(m) + c \log(1/\delta).$$

With $\mathcal{H}$ the set of linear hyperplanes, “Optimal” only in the sense that it optimizes the particular $\psi$ function used. Does not minimize the probability of error (the expected risk) — cf. Bayes point machine.

Is $\mathcal{A}_{MM}$ an instance of $\mathcal{A}_{srm}^{\mathcal{H},\delta}$?
Data Dependent SRM: $A^{\mathcal{H}(z), \delta}_{\text{dsrm}}$

New algorithm: *Penalize complexity of $\mathcal{H}(z)$ as if it is independent of $z$. Consider $\mathcal{H}(z) = (\mathcal{H}_i(z))_i$ such that*

$$\log \mathcal{N}(\epsilon, \mathcal{H}_i(z), \ell_1(z)) \leq \phi(i)$$

where the dependence of $\mathcal{H}_i(z)$ on $z$ is “mild”. Suppose for data-independent $\mathcal{H}_i$ and $h_{\text{emp}} = A^{\mathcal{H}_i}_{\text{erm}}(z)$ have a bound

$$R(h_{\text{emp}}) \leq \psi(R_{\text{emp}}(h_{\text{emp}}, z), \phi(i)) =: \chi(i)$$

Let

$$i^* := \arg \min_i \chi(i) \quad A^{\mathcal{H}(z), \delta}_{\text{dsrm}}(z) := A^{\mathcal{H}_{i^*}(z)}_{\text{erm}}(z)$$

Gist: penalize complexity ignoring data-dependence; apply SRM.

Problem: how to avoid crazy situations as on previous page?
How to conceptualize what’s going on?

Can not strictly justify the algorithm as an application of SRM.

Can use specific reasoning to give high probability bounds on performance.

How to generalize the results. What is the intuitive idea?

If as well as $R_{\text{emp}}(h, z) = 0$ we have $\gamma_z(h) = \gamma \gg 0$, then $R(h)$ is small.

We are *lucky* if our data is like this.

Would like to capture this notion in a general, formal way.
Luckiness

The margin $\gamma_z(h_w)$ measures how lucky $h_w$ is on $z$. In general $L : \mathcal{H} \times \mathcal{Z}^m \to \mathbb{R}$.

Would like a bound that says with probability $1 - \delta$ over a random draw of $z$ according to $\mathbb{P}_z$ if $R_{\text{emp}}(h, z) = 0$ and $\omega(L(h, z), \delta) \leq 2^d$ then

$$R(h) \leq \psi(m, d) \quad \diamond$$

The parameter $d$ is an effective complexity.

There needs to be some restrictions on $L$: if we “use up” all of the information in the sample estimating its luckiness there is “none left” to estimate $R(h)$.

A bound like $\diamond$ is a bound for the algorithm

$$\mathcal{A}_{\text{lucky}}^{\mathcal{H}, L} := z \mapsto \arg \min_{h \in \mathcal{H}} \psi(m, \log (\omega(L(h, z), \delta)))$$
Luckiness (continued)

Just for binary classification. Given a luckiness function $L: \mathcal{H} \times \mathcal{Z}^m \to \mathbb{R}$, the level is

$$\ell_L(h, z) := \left| \left\{ (l(g(x_i), y_i))_{i=1}^m : L(g, z) \geq L(h, z) \right\} \right|,$$

the number of dichotomies induced on $z$ by functions at least as lucky as $h$. Require $L$ to be well behaved:

$L$ is probably smooth w.r.t. $\omega: \mathbb{R} \times [0,1] \to \mathbb{N}$ if for all $m \in \mathbb{N}$ all distributions $\mathbb{P}, \mathbb{Z}$ of size $2m$, and all $\delta \in [0,1]$

$$\mathbb{P}_{\mathbb{Z}}(\exists h \in \mathcal{H}: \ell_L(h, \mathbb{Z}_{[1:2m]}) > \omega(L(h, \mathbb{Z}_{[1:m]})), \delta)) \leq \delta.$$  

If $L$ is probably smooth w.r.t. $\omega$, $\delta = (\delta_i)_i$, $\sum_i \delta_i = \delta$, with probability $1 - \delta$ over a random draw of $z$, if $R_{\text{emp}}(h, z) = 0$ and $\omega(L(h, z), \delta_d/4) \leq 2^d$ then

$$R(h) \leq \frac{2}{m} \left( d + \log_2(4/\delta_d) \right)$$  

Effectively $\mathcal{H}(z) = (\mathcal{H}_i(z))_i$ with $\mathcal{H}_i(z) = \{ h \in \mathcal{H}: \omega(L(h, z), \delta_i/4) \leq 2^i \}$.  

Robert C Williamson: SRM and VC Theory (Statistical Learning Theory) — Dagstuhl, July 2001 23
Comments on Luckiness

- Can put $A_{MM}$ into this framework, but need a more general notion of probable smoothness and a more complicated luckiness bound.

- The luckiness function $L$ is how we encode our prior knowledge. We weight with $\delta_i$ the $i$th data-dependent hypothesis class

$$\mathcal{H}_i(z) = \{ h \in \mathcal{H} : \omega(L(h, z), \delta_i/4) \leq 2^i \}$$

- Key practical difficulty is showing $L$ is probably smooth w.r.t. a “good” $\omega$ — the smaller the $\omega$ the tighter $\heartsuit$ is.

- Perhaps key value is conceptual. Shows one can rigorously study $A_{\mathcal{H},\delta}$.
Problems with Luckiness

The luckiness bound also holds for the algorithm

\[ A_{\text{worst-lucky}}^{L,H} := \arg \max_{h \in S(H, L, z)} R(h) \]

where

\[ S(H, L, z) = \left\{ h \in H : R_{\text{emp}}(h, z) = 0 \text{ and } L(h, z) = L \left( A_{\text{lucky}}^{H, L}(z), z \right) \right\} \]

Could argue that this is not such a huge problem in the sense that all it is saying is that we have codified our “prior knowledge” into the luckiness function and the bound holds for the worst situation consistent with what we have observed and our prior structure.

Absence of characterization results for any algorithms other than \( A_{\text{erm}}^H \)
Adherents of the various non-NPW schools take advantage of “lucky observations” to make more conclusive sounding statements than they would for “unlucky outcomes”.

— Jack Kieffer: Conditional Confidence Statements ... 1977

The development from $\mathcal{A}_{\text{erm}}$ through $\mathcal{A}_{\text{srm}}$ to $\mathcal{A}_{\text{dsrm}}$ by way of the luckiness framework has an analogy with the notions of conditional confidence in classical statistics.

Traditional confidence intervals give a “guarantee” that does not condition on the data. This is often unsatisfactory — if the data is lucky you should exploit it.

Deal with standard CI difficulty: after the experiment you can not say that with high probability, the true parameter lies in the interval...

There are several different competing approaches to this topic with hundreds of pages written. There is extremely little though on applying the ideas to the sort of problems studied in the machine learning community.
Refinements

Alternate complexity Measures e.g. Rademacher averages

Refined Sample Complexity Bounds Lots of work devoted to improving the bounds

Cross-Validation etc State as an algorithm?

Other All sorts of refinements to the standard reasoning such as error shells, dealing with dependent observations, non-bounded losses etc.
Other Learning Theories

**Bayesian** Alternate “inductive principle” — difficult to theoretically analyse expected risk

**PAC-Bayesian** Allows one to analyse algorithms motivated by Bayesian reasoning in a frequentist manner

**Algorithmic Stability** Bound expected risk in terms of stability properties of algorithms

**Compression Framework** Determine mistake bounds for online algorithms. Can convert to batch algorithms.
A new approach: *Algorithmic Luckiness*. Idea is to study the algorithm more directly. Luckiness is now a function

\[ L(\mathcal{A}, z) \]

of the *algorithm* \(\mathcal{A}\) and the data \(z\). Formal definition via permutations of the data sample to link in with the permutation trick.

Allows one to get results similar to those for classical VC theory, compression schemes, algorithmic stability and margin. See Ralf’s talk.

And the winner is . . . Lucky!
— Britney Spears: Lucky (2000)
Conclusions

“Complex theories do not work, simple algorithms do.”

Complex theories explain the work that simple algorithms do.

“Learning theory has one clear goal: to understand the phenomenon of induction that exists in nature.”

Learning theory has one clear goal: to understand the phenomenon of good performance of induction algorithms in order to be able to design better ones.

Worth focusing on the questions rather than the techniques.
Some Questions

1. Is an inductive principle anything more than just an algorithm with knobs on it?

2. Is it “merely” a way of encoding prior information? What other ways of encoding prior information in an intuitively nice form are there?

3. Extend luckiness type reasoning to a wider range of algorithms (avoiding variations of $A_{\text{worst}}$).

4. Develop a better (more “natural”) notion of non-trivial consistency.

5. Understand to what extent union bound looseness is unavoidable.

6. Formally relate SRM type theory to conditional confidence procedures.

7. Unify different approaches to bounding the performance of an algorithm (e.g. PAC bayesian, online mistake bounds, algorithmic stability).
The Big Question

**Characterize** what makes a good algorithm, and so develop better ones.
Who?

Stefan Banach  Thomas Bayes  Sergey Bernstein  George Boole
Who else?

Francesco Cantelli  Pavnutii Chebyshev  Hermann Chernoff  Carl Gauss
Who else?

David Hilbert    Wassily Hoeffding    Harold Jeffreys    Jack Kieffer
Who else?

Andrei Kolmogorov  Jerzy Neyman  Richard Price  Claude Shannon
Thanks to

[Image of a person]