Limitations of Kernel and Multiple Kernel Learning

John Shawe-Taylor
Centre for Computational Statistics
and Machine Learning
Department of Computer Science
UCL Engineering
University College London
jst@cs.ucl.ac.uk

Including work with Zak Hussain

Relation to SIMBAD

- When can we learn using Euclidean representations?
- Question can be asked at two levels:
  - Is the data naturally separable in the given representation and if not can we adjust the representation to ensure it is?
  - For a given set of classifiers H over inputs X can we embed X into Hilbert space so that H subset of linear threshold functions?
- We consider this second question.

Context

- If we ignore the problem of how to arrive at the representation, it is clear that we can represent any classifier

$$f : X \longrightarrow \{-1, 1\}$$

as a (large margin) linear threshold by simply choosing the embedding:

$$\phi : x \in X \longmapsto f(x) \in \mathbb{R}$$

- Now classifier $y = \mathrm{sgn}(\phi(x))$ gives perfect classification.
- What about if we have a set of classifiers - is there an embedding good for all of them?

Impossibility of representing classes of functions

- Surprisingly there are many classes that are known to be learnable for which no such embedding exists.
- Ben-David, Eiron and Simon (2002) show that, for VC classes with VC dimension d on m inputs, only a vanishing fraction can be embedded into a Euclidean space of dimension much less than m meaning learning is not possible with this representation.
- Note that finite VC dimension d means learnable with an error bound scaling as $\sqrt{d/m}$.
- Proof uses a counting argument – there are too many VC classes and/or too few linear threshold classes.

What about Support Vector Machines?

- Support vector machines (SVMs) work in very (can be infinite) high dimensional spaces, so no problem?
- SVMs overcome 'curse of dimensionality' by maximising the margin γ, eg bounds have the form:

$$\begin{aligned} P(y \neq \operatorname{sgn}(g(\mathbf{x}))) &= \mathbb{E}\left[\mathcal{H}(-yg(\mathbf{x}))\right] \\ &\leq \frac{1}{m\gamma}\sum_{i=1}^{m}\xi_i + \frac{2}{m\gamma}\sqrt{\sum_{i=1}^{m}\kappa(\mathbf{x}_i,\mathbf{x}_i)} + 3\sqrt{\frac{\ln(2/\delta)}{2m}} \end{aligned}$$

- Note that for the Gaussian kernel this reduces to

$$P(y \neq \operatorname{sgn}(g(\mathbf{x}))) \leq \frac{1}{m\gamma}\sum_{i=1}^{m}\xi_i + \frac{2}{\sqrt{m}\gamma} + 3\sqrt{\frac{\ln(2/\delta)}{2m}}$$

Random projections

- There is an intimate connection between the effect of having a large margin and resilience to random projects (Balcan, Blum & Vempala, 2004):
  - bounds on large margin classification can be obtained by showing that random projections into a low dimensional ($O(R^2/\gamma^2)$) space ensures still have good linear separability
  - hence same non-representability applies
- Result already obtained by Ben-David et al is counter-intuitive, but very powerful: for the overwhelming majority of classes that can be learnt there is no kernel that will render them learnable by SVMs.
- One weakness is that it is only an existence proof – would like to see one of these classes.

Concrete example

- Forster et al (2002) significantly advanced the technology of analysing this problem and showed a concrete example.
- If we consider the sign matrix M of concepts – m rows indexed by examples, n columns by classifiers, entries $+1, -1$, the minimal dimension d to represent this is lower bounded by

$$d \geq \max\left\{\frac{\sqrt{mn}}{\|M\|}, mn\left(\sum_{i=1}^{d}\sigma_i(M)\right)^{-1}\right\}$$

Concrete example

- Has been applied to a few concrete function classes
- Monomials over n boolean variables are all functions representable as conjunctions of literals (variables or their negations)
- This class can only be embedded with a margin of at most $1/\sqrt{n}$.
- Unfortunately, does not make learning more difficult, since VC dimension of monomials is n.

Converting learning to convex optimisation

- The aim of this talk is to revisit this impasse by turning to an alternative large margin approach
- Maximising the margin while controlling the 1-norm of the weight vector gives a form of boosting
- Can also be combined with the SVM approach
- First will need to take a closer look at the error bounds

Main Rademacher theorem

The main theorem of Rademacher complexity: with probability at least $1 - \delta$ over random samples S of size m, every $f \in \mathcal{F}$ satisfies

$$\mathbb{E}\left[f(\mathbf{z})\right] \leq \hat{\mathbb{E}}\left[f(\mathbf{z})\right] + R_m(\mathcal{F}) + \sqrt{\frac{\ln(1/\delta)}{2m}}$$

where $R_m(\mathcal{F})$ is the Rademacher complexity of $\mathcal{F}$

$$R_m(\mathcal{F}) = \mathbb{E}_{\mathcal{D}^m}\mathbb{E}_{\sigma}\left[\sup_{f\in\mathcal{F}} \frac{2}{m}\sum_{i=1}^{m}\sigma_i f\left(\mathbf{z}_i\right)\right]$$

Empirical Rademacher theorem

- Since the empirical Rademacher complexity

$$\hat{R}_m(\mathcal{F}) = \mathbb{E}_\sigma \left[\sup_{f \in \mathcal{F}} \frac{2}{m} \sum_{i=1}^{m} \sigma_i f(\mathbf{z}_i) \middle| \mathbf{z}_1, \ldots, \mathbf{z}_m \right]$$

is concentrated, we can make an application of McDiarmid's theorem to obtain with probability at least $1 - \delta$

$$\mathbb{E}_{\mathcal{D}}[f(\mathbf{z})] \leq \hat{\mathbb{E}}[f(\mathbf{z})] + \hat{R}_m(\mathcal{F}) + 3\sqrt{\frac{\ln(2/\delta)}{2m}}.$$

McDiarmid's inequality

Theorem

Let $X_1, \ldots, X_n$ be independent random variables taking values in a set A, and assume that $f : A^n \to \mathbb{R}$ satisfies

$$\sup_{x_1,\ldots,x_n,\hat{x}_i \in A} |f(x_1,\ldots,x_n) - f(x_1,\ldots,\hat{x}_i, x_{i+1},\ldots,x_n)| \leq c_i,$$

for $1 \leq i \leq n$. Then for all $\epsilon > 0$,

$$P\left\{f\left(X_1,\ldots,X_n\right) - \mathbb{E}f\left(X_1,\ldots,X_n\right) \geq \epsilon\right\} \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^n c_i^2}\right)$$

- Hoeffding is a special case when $f(x_1,\ldots,x_n) = S_n$

Application to large margin classification

- Rademacher complexity comes into its own for Boosting and SVMs.
- SVM bound we have already seen – now investigate boosting as will enable conversion of VC classes into a convex margin optimisation

Application to Boosting

- We can view Boosting as seeking a function from the class (H is the set of weak learners)

$$\left\{ \sum_{h \in H} a_h h(\mathbf{x}) : a_h \geq 0, \sum_{h \in H} a_h \leq B \right\} = \mathrm{conv}_B(H)$$

  by minimising some function of the margin distribution.
- Adaboost corresponds to optimising an exponential function of the margin over this set of functions.
- We will see how to include the margin in a moment, but concentrate on computing the Rademacher complexity now.

Rademacher complexity of convex hulls

Rademacher complexity has a very nice property for convex hull classes:

$$
\begin{aligned}
\hat{R}_m(\mathrm{conv}_B(H)) &= \frac{2}{m}\mathbb{E}_\sigma\left[\sup_{h_j\in H,\sum_j a_j\le B}\sum_{i=1}^m \sigma_i \sum_j a_j h_j(\mathbf{x}_i)\right] \\
&\le \frac{2}{m}\mathbb{E}_\sigma\left[\sup_{h_j\in H,\sum_j a_j\le B}\sum_j a_j \sum_{i=1}^m \sigma_i h_j(\mathbf{x}_i)\right] \\
&\le \frac{2}{m}\mathbb{E}_\sigma\left[\sup_{h_j\in H} B\sum_{i=1}^m \sigma_i h_j(\mathbf{x}_i)\right] \le B\hat{R}_m(H).
\end{aligned}
$$

Rademacher complexity of convex hulls cont.

- Hence, we can move to the convex hull without incurring any complexity penalty for $B = 1$!
- Margin is incorporated by applying Rademacher theorem to class with piecewise linear loss function $\mathcal{A}$:
  - loss zero if margin bigger than γ
  - linearly increasing loss with slope $1/\gamma$ as margin decreases from γ to 0
  - loss 1 for negative margin $\equiv$ misclassification

Final Boosting bound

- Gives bound:

$$P(y \neq \mathrm{sgn}(g(\mathbf{x}))) \quad = \quad \mathbb{E}\left[\mathcal{H}(-yg(\mathbf{x}))\right] \leq \mathbb{E}\left[\mathcal{A}(-yg(\mathbf{x}))\right]$$
$$\leq \frac{1}{m}\sum_{i=1}^{m}\xi_i + \hat{R}(H)\sum_h a_h + 3\sqrt{\frac{\ln(2/\delta)}{2m}}$$

where $\xi_i = \left(1 - y_i \sum_h a_h h(\mathbf{x}_i)\right)_+$ are the so-called slack variables.

Linear programme

- Converting into a corresponding optimisation with the 1-norm of the slack variables we arrive at Linear programming boosting that minimises

$$\sum_h a_h + C \sum_{i=1}^{m} \xi_i,$$

where $\xi_i = (1 - y_i \sum_h a_h h(\mathbf{x}_i))_+$.

Linear programming boosting

- Can view $h(\mathbf{x}_i)$ as a set $\mathbf{H}_{ij}$ of 'weak' learners with j indexing the set (and include the constant function as one weak learner and negative of each weak learner):

$$\begin{array}{ll} \min_{\mathbf{a},\xi} & \|\mathbf{a}\|_1 + C\sum_{i=1}^m \xi_i \\ \text{subject to} & y_i \mathbf{H}_i \mathbf{a} \geq 1 - \xi_i,\ \xi_i \geq 0,\ a_i \geq 0 \\ & i = 1, \dots, m. \end{array}$$

Alternative version

- Can equivalently explicitly optimise margin γ with 1-norm fixed:

$$\begin{array}{ll} \max_{\gamma,\mathbf{a},\xi} & \gamma - D\sum_{i=1}^{m}\xi_i \\ \text{subject to} & y_i\mathbf{H}_i\mathbf{a} \geq \gamma - \xi_i,\ \xi_i \geq 0, a_j \geq 0 \\ & \sum_{j=1}^{N} a_j = 1. \end{array}$$

- Dual has the following form:

$$\begin{array}{ll} \min_{\beta,\mathbf{u}} & \beta \\ \text{subject to} & \sum_{i=1}^{m} u_i y_i \mathbf{H}_{ij} \leq \beta,\ j = 1,\ldots,N, \\ & \sum_{i=1}^{m} u_i = 1,\ 0 \leq u_i \leq D. \end{array}$$

Column generation

Can solve the dual linear programme using an iterative method:

1. initialise $u_i = 1/m, i = 1, \ldots, m$, $\beta = \infty$, $J = \emptyset$
2. choose $j^\star$ that maximises $f(j) = \sum_{i=1}^{m} u_i y_i \mathbf{H}_{ij}$
3. if $f(j^\star) \leq \beta$ solve primal restricted to J and exit
4. $J = J \cup \{j^\star\}$
5. Solve dual restricted to set J to give u_i, β
6. Go to 2

LP Boost

- Note that u_i is a distribution on the examples
- Each j added acts like an additional weak learner
- $f(j)$ is simply the weighted classification accuracy
- Hence gives 'boosting' algorithm - with previous weights updated satisfying error bound
- Guaranteed convergence and soft stopping criteria

Implementing VC class through LP Boost

- Each classifier h is a weak learner through the mapping:

$$h : \mathbf{x} \longmapsto h(\mathbf{x}) \in \mathbb{R}$$

- If we restrict to a finite set of m training data, Sauer's lemma tells us there are at most

$$\left(\frac{em}{d}\right)^d$$

  distinct classifiers in the class, so obtain Linear programme with polynomially many constraints
- Actually have potentially extended the class of functions as linear combinations are also allowed
- Have ducked the problem of how to index the functions, but would follow from learning algorithm for VC class

Between Boosting and SVMs

- Can we move between boosting and SVMs to ameliorate the problem with explicitly enumerating all of the classifiers as constraints?
- Multiple kernel learning aims to combine a number of different kernels and select a subset of them as part of the training
- Standard multiple kernel learning uses the optimisation:

$$\begin{array}{ll} \min_{\mathbf{w}_t, b, \gamma, \xi} & \left(\sum_{t=1}^{N} \|\mathbf{w}_k\|_2\right)^2 + C \sum_{i=1}^{m} \xi_i \\ \text{subject to} & y_i \left(\sum_{t=1}^{N} \langle \mathbf{w}_t, \phi_t(\mathbf{x}_i) \rangle + b\right) \geq \gamma - \xi_i,\ \xi_i \geq 0. \end{array}$$

Multiple kernel learning

- Equivalently MKL puts a 1-norm constraint on a linear combination of kernels:

$$\left\{ \kappa(\mathbf{x}, \mathbf{z}) = \sum_{t=1}^{N} z_t \kappa_t(\mathbf{x}, \mathbf{z}) : z_t \geq 0, \sum_{t=1}^{N} z_t = 1 \right\}$$

and trains an SVM while optimizing z_t – a convex problem, c.f. group Lasso.

- This is equivalent to performing Linear programming boosting with the weak learners:

$$\mathcal{H} = \bigcup_{t=1}^{N} \mathcal{F}_t \quad \text{where} \quad \mathcal{F}_t = \{\mathbf{x} \mapsto \langle \mathbf{w}, \phi_t(\mathbf{x}) \rangle : \|\mathbf{w}\|_2 \leq 1\}$$

Multiple kernel learning via LP Boost

- To implement MKL as an LP Boosting we need to choose the weak learner h that maximises

$$\max_{h \in \mathcal{H}} \sum_{i=1}^{m} u_i y_i h(\mathbf{x}_i)$$

- But we can optimise over $\mathcal{F}_t$ since

$$\max_{\|\mathbf{w}\| \leq 1} \sum_{i=1}^{m} u_i y_i \langle \mathbf{w}, \phi_t(\mathbf{x}_i) \rangle = \left\langle \mathbf{w}, \sum_{i=1}^{m} u_i y_i \phi_t(\mathbf{x}_i) \right\rangle$$

which is maximised by choosing $\mathbf{w}$ parallel to

$$\sum_{i=1}^{m} u_i y_i \phi_t(\mathbf{x}_i)$$

Multiple kernel learning

- We obtain a bound on generalisation:

$$P(y \neq \operatorname{sgn}(g(\mathbf{x})))$$
$$\leq \frac{1}{m\gamma}\sum_{i=1}^{m}\xi_i + \frac{1}{\gamma}\hat{R}_m\left(\bigcup_{t=1}^{N}\mathcal{F}_t\right) + 3\sqrt{\frac{\ln(2/\delta)}{2m}}$$

- but are missing a bound on the Rademacher complexity of the class of weak learners.

Bounding MKL

- First note further applications of McDiarmid gives with probability $1 - \delta_0$ of a random selection of σ^*:

$$\hat{R}_m(\mathcal{H}) \leq \frac{2}{m} \sup_{f \in \mathcal{H}} \sum_{i=1}^{m} \sigma_i^* f(\mathbf{x}_i) + 4\sqrt{\frac{\ln(1/\delta_t)}{2m}}$$

and

$$\frac{2}{m} \sup_{f \in \mathcal{F}_t} \sum_{i=1}^{m} \sigma_i^* f(\mathbf{x}_i) \leq \hat{R}_m(\mathcal{F}_t) + 4\sqrt{\frac{\ln(1/\delta_t)}{2m}}$$

with probability $1 - \delta_t$

Bounding MKL

- Hence taking $\delta_t = \delta/2(N+1)$ for $t = 0, \ldots, N$

$$\hat{R}_m\left(\mathcal{H} = \bigcup_{t=1}^{N} \mathcal{F}_t\right)$$

$$\leq \frac{2}{m} \sup_{f \in \mathcal{F}} \sum_{i=1}^{m} \sigma_i^* f(\mathbf{x}_i) + 4\sqrt{\frac{\ln(2(N+1)/\delta)}{2m}}$$

$$\leq \frac{2}{m} \max_{1 \leq t \leq N} \sup_{f \in \mathcal{F}_t} \sum_{i=1}^{m} \sigma_i^* f(\mathbf{x}_i) + 4\sqrt{\frac{\ln(2(N+1)/\delta)}{2m}}$$

$$\leq \max_{1 \leq t \leq N} \hat{R}_m(\mathcal{F}_t) + 8\sqrt{\frac{\ln(2(N+1)/\delta)}{2m}}$$

with probability $1 - \delta/2$.

Concluding remarks

- Examined the limitations of learning with linear function classes
- Reviewed negative results for SVMs when compared to general VC classes
- Showed how using LP boosting this problem can be overcome
- Extended the approach to Multiple kernel learning that may enable a less explicit enumeration of the functions