9.520: Class 11

Multiclass Classification

Ryan Rifkin

“It is a tale
Told by an idiot, full of sound and fury,
Signifying nothing.”

Macbeth, Act V, Scene V

What Is Multiclass Classification?

Each training point belongs to one of N different classes. The goal is to construct a function which, given a new data point, will correctly predict the class to which the new point belongs.

What Isn't Multiclass Classification?

There are many scenarios in which there are multiple categories to which points belong, but a given point can belong to multiple categories. In its most basic form, this problem decomposes trivially into a set of *unlinked* binary problems, which can be solved naturally using our techniques for binary classification.

A First Idea

Suppose we knew the density, $p_i(\mathbf{x})$, for each of the N classes. Then, we would predict using

$$f(\mathbf{x}) = \text{arg} \max_{i \in 1,\ldots,N} p_i(\mathbf{x}).$$

Of course we don't know the densities, but we could estimate them using classical techniques.

The Problem With Densities, and Motivation

Estimating densities is hard, especially in high dimensions with limited data.

For binary classification tasks, we have seen that directly estimating a smooth separating function gives better results than density estimation (SVM, RLSC). Can we extend these approaches usefully to the multiclass scenario?

A Simple Idea — One-vs-All Classification

Pick a good technique for building binary classifiers (e.g., RLSC, SVM). Build N different binary classifiers. For the ith classifier, let the positive examples be all the points in class i, and let the negative examples be all the points not in class i. Let f_i be the ith classifier. Classify a new function using

$$f(\mathbf{x}) = \arg\max_i f_i(\mathbf{x}).$$

The Truth

To the best of my knowledge, choosing properly tuned regularization classifiers (RLSC, SVM) as your underlying binary classifiers and using one-vs-all (OVA) works as well as anything else you can do.

If you actually have to solve a multiclass problem, I strongly urge you to simply use OVA and not worry about anything else.

Other Approaches

There have been two basic approaches to extending regularization ideas to multiclass classification:

- "Single Machine" approaches — try to solve a *single* optimization problem that trains many binary classifiers simultaneously.

- "Error Correcting Code" approaches — try to combine binary classifiers in a way that lets you exploit decorrelations and correct errors.

These approaches are not completely exclusive.

Weston and Watkins, Vapnik

The first "single machine" approach:

$$\begin{array}{rl} \min\limits_{\mathbf{f_1},\ldots,\mathbf{f_N}\in\mathcal{H},\xi\in\mathbf{R}^{\ell(\mathbf{N}-\mathbf{1})}} & \sum_{i=1}^{N} ||f_i||_K^2 + C\sum_{i=1}^{\ell}\sum_{j\neq y_i}\xi_{ij} \\ \text{subject to}: & f_{y_i}(\mathbf{x_i}) + b_{y_i} \geq f_j(\mathbf{x_i}) + b_j + 2 - \xi_{ij} \\ & \xi_{ij} \geq 0 \end{array}$$

Key idea. Suppose that point i is in class y_i. Then, for $j \neq y_i$, we want (abusing our notation w.r.t. b),

$$f_{y_i}(\mathbf{x_i}) - f_j(\mathbf{x_i}) \geq 2,$$

or we pay a linear penalty of ξ_{ij}.

WW Analysis I

This idea seems intuitively reasonable. Is it good?

Weston and Watkins perform experiments. On 2 out of 5 datasets, they find that their approach performs substantially better than OVA, and about the same on the rest. However, they state that "to enable comparison, for each algorithm $C = \infty$ was chosen (the training data must be classified without error)," so they are performing ERM, not regularization ($C = \infty \Longleftrightarrow \lambda = 0$). A Gaussian kernel was used, with σ the same for each method (not necessarily a good idea), and no information about how this σ was chosen.

WW Analysis II

Under what circumstances would we expect this method to outperform a OVA approach? Tough to say. We'd need a situation where it would be hard to actually separate the data, but where there exist meaningful subsets of the data wwere even though we can't assign a positive value to the correct class, we can assign a less negative value to it than other classes. Or, we need subsets where even though we're assigning positive values to some incorrect class, we're assigning even more strongly positive values to the correct classes.

Challenge: Come up with a set of examples that actually has this property. Double challenge: Have it hold when the kernel is a properly-tuned Gaussian.

WW Analysis III

There is no evidence that this scheme is any more accurate than an OVA scheme. It is, however, much slower. Instead of N problems of size ℓ, we need to solve a problem of size $(N-1)\ell$. Moreover, although the authors derive a dual problem, they don't show any way to decompose it; the resulting problem is much more complex than a standard SVM.

Lee, Lin and Wahba: Motivation

In an earlier paper, Lin showed that asymptotically, as we let $\ell \to \infty$ and $\lambda \to 0$, the solution of an SVM will tend to

$$f(\mathbf{x}) = \text{sign}\left(p(\mathbf{x}) - \frac{1}{2}\right).$$

This is good for binary classification.

Now consider multiclass classification with an OVA scheme. In regions where there is a dominant class i for which $p(\mathbf{x}) > \frac{1}{2}$, all is good. If there isn't, then all N of the OVA functions will return -1, and we will be unable to recover the most likely class.

Lee, Lin and Wahba: Notation

For $i \in 1, \ldots, N$, define v_i as the N dimensional vector with a 1 in the ith coordinate and $\frac{1}{N-1}$ elsewhere. The vector v_i serves as the "target" for points in class i — we try to get function outputs that are very close to the entries of v_i.

Given a training point $\mathbf{x_i}$, we try to make $f_j = -\frac{1}{N-1}$ for $j \neq y_i$, and then we also require that $\sum_{j=1}^{N} f(\mathbf{x_i}) = 0$. This leads us to...

Lee, Lin and Wahba: Formulation

$$\begin{array}{lcr} \min\limits_{f_1,\ldots,f_N \in \mathcal{H}_K} & \frac{1}{\ell}\sum_{i=1}^{\ell}\sum_{j=1, j\neq y_i}^{N}(f_j(\mathbf{x}_i) + \frac{1}{N-1})_+ + \lambda \sum_{j=1}^{C} ||f_j||_K^2 & \\ \text{subject to :} & \sum_{j=1}^{C} f_j(\mathbf{x}) = 0 & \forall \mathbf{x} \end{array}$$

Lee, Lin and Wahba: Analysis

Asymptotically, this formulation gives

$$f_i(\mathbf{x}) = \begin{cases} 1 & \text{iff } i = \arg\max p_j(\mathbf{x}) \\ -\frac{1}{N-1} & \text{otherwise} \end{cases}$$

In turn, this means that our final multiclass classification function will be

$$f(\mathbf{x}) = \arg\max p_j(\mathbf{x}).$$

Lee, Lin and Wahba: Issues I

- Even under the conditions of the analysis, this produces a different result from OVA only when $\arg\max p_j(\mathbf{x}) < \frac{1}{2}$ — the Bayes error rate must be $> \frac{1}{2}$, indicating a very tough problem that should likely be modelled differently.

- The "problem" with an OVA-SVM scheme relies on the linearity of the SVM loss. If we use instead RLSC or a square-loss SVM, these problems disappear, and the Bayes rule is again recovered asymptotically.

Lee, Lin and Wahba: Issues II

- Like the WW formulation, this formulation is big, and no decomposition method is provided.

- This is an *asymptotic* analysis. It requires $n \to \infty$ and $\lambda \to 0$, and no rates are provided. But *asymptotically*, density estimation will allow us to recover the optimal Bayes rule. The burden is on the authors to show that there is a useful middle ground where this thing performs well.

Lee, Lin and Wahba: Experiments

Two toy examples. In one, no comparison is made to other techniques. In the other, they compare to OVA. The data is 200 points in one-dimension in thre classes, constructed so that class 2 never has conditional probability $> 50\%$. The LLW approach predicts class 2 in the region where it's more likely than any other class (total error rate .389, and the OVA system fails there (total error .4243). I cannot understand how there parameters were chosen.

Crammer and Singer: Formulation

They consider a formulation that is a modified version of WW:

$$\begin{array}{rc} \min\limits_{\mathbf{f_1},\ldots,\mathbf{f_N}\in\mathcal{H},\xi\in\mathbf{R}^\ell} & \sum_{i=1}^{N} ||f_i||_K^2 + C\sum_{i=1}^{\ell}\sum_{j\neq y_i}\xi_i \\ \text{subject to}: & f_{y_i}(\mathbf{x_i}) \geq f_j(\mathbf{x_i}) + 1 - \xi_i \\ & \xi_i \geq 0 \end{array}$$

Key difference: there is only one slack variable ξ_i per data point, rather than $N-1$ slack variables per data point as in WW. Instead of paying for *each* class $j \neq i$ for which $f_i(\mathbf{x}) < f_j(\mathbf{x}) + 1$, they pay only for the *largest* $f_j(\mathbf{x})$.

Crammer and Singer: Development

The majority of the C&S paper is devoted to efficiently solving the formulation. The Lagrangian dual is taken, dot products are replaced with kernel products, and a dual decomposition algorithm is developed. This algorithm is substantially more complicated than the SVM algortihm; the mathematics is fairly involved, but elegant.

In the experimental section of their paper, C&S make claims as to both the speed and the accuracy of their method. . .

Crammer and Singer: Speed

C&S claim that their method is orders of magnitude faster than than an OVA approach. However:

- They benchmark the OVA approach by using Platt's 1998 results, in which he implemented SMO with no caching of kernel products. In contrast, their system used 2 GB of RAM to cache kernel products.

- The paper states that the fastest version has "two more technical improvements which are not discussed here but will be documented in the code that we will shortly make available"; as of this writing, two years after the paper was submitted to the Journal of Machine Learning Research, the code has not been made available

Crammer and Singer: Accuracy

C&S consider data sets from the UCI repository. Their chart shows the difference in error rates between their system and an OVA system, *but not the actual error rates*.

The largest differences appeared to be on the `satimage` (difference of approximately 6.5%) and `shuttle` (difference of approximately 3%) data sets. Through personal communication, Crammer indicated that the actual error rates for his system on these two data sets were 8.1% and 0.1%, respectively. In my own one-vs-all experiments on these data sets, I achieved error rates of 7.9% for the `satimage` data and 0.35% on the `shuttle` data. These differences are *much* smaller than the differences reported by C&S.

Dietterich and Bakiri: Introducing the ECC Approach

Consider a $\{-1, 1\}$-valued matrix M of size N by F where F is the number of classifiers to be trained. The ith column of the matrix induces a partition of the classes into two "metaclasses", where a point $\mathbf{x_i}$ is placed in the positive metaclass for the jth classifier if and only if $M_{y_i j} = 1$.

When faced with a new test point $\mathbf{x}$, we compute $f_1(\mathbf{x}), \ldots, f_F(\mathbf{x})$, take the signs of these values, and then compare the Hamming distance between the resulting vector and each row of the matrix, choosing the minimizer:

$$f(\mathbf{x}) = \arg \min_{r \in 1,\ldots,N} \sum_{i=1}^{F} \left(\frac{1 - \mathrm{sign}(M_{ri} f_i(\mathbf{x}))}{2} \right).$$

Dietterich and Bakiri: Introducing the ECC Approach

D&B suggest that M be constructed to have good *error-correcting* properties — if the minimum Hamming distance between rows of M is d, then the resulting multiclass classification will be able to correct any $\lfloor \frac{d-1}{2} \rfloor$ errors.

In learning, good *column* separation is also important — two identical (or opposite) columns will make identical errors. This highlights the key difference between the multiclass machine learning framework and standard error-correcting code applications.

Dietterich and Bakiri: Experiments

A large number of experiments were performed. Decision trees and feed-forward neural networks were used as base learners, and several methods of generating codes were tried.

In general, it seems that their approach outperformed OVA approaches. However, differences were often small, the quality of the underlying binary classifiers is unknown, and often only relative performance results are given, so this work is difficult to evaluate.

Allwein, Schapere and Singer: Generalizing D&B

AS&S generalize D&B in two main ways:

- Allow 0-entries (in addition to 1 and -1) in the M matrix — if $M_{ij} = 0$, then examples from class i are simply not used in classifier j.

- Use *loss-based decoding* to classify examples — instead of taking the sign of the output of each classifier, compute the actual loss, using the training loss function (hinge loss for SVM, square loss for RLSC).

AS&S observe that OVA approaches, "all-pairs" approaches, the ECC approach of D&B, and generalizations of this to include 0 in the matrix all fit into their framework.

Allwein, Schapere and Singer: Experimental Setup

AS&S tested multiclass SVMs and AdaBoost using five different matrices:

- OVA and AVA: one-vs-all and "all-pairs"

- COMPLETE: $O(2^N)$ columns

- DENSE and SPARSE: Randomized codes with and without zeros in M.

Their conclusion: "For SVM, it is clear that the widely used one-against-all code is inferior to all the other codes we tested."

Allwein, Schapere and Singer: Experimental Conditions

AS&S performed all SVM experiments using a polynomial kernel of degree 4; no justification for this choice is made. No information about regularization parameters is given. Fortunately, AS&S gave actual numerical results for their experiments.

The difference in performance was large on only two data sets: `yeast` and `satimage`. I performed my own experiments on these data sets, using Gaussian kernels. The Guassian parameter σ was tuned by "cheating" on the test set, although the accuracy rates were not very sensitive to choice of σ (and see later).

Allwein, Schapere and Singer: Experimental Comparison

	OVA	AVA	COM	DEN	SPA
Allwein et al.	40.9	27.8	13.9	14.3	13.3
Rifkin	7.9	7.9	8.0	8.2	8.2

Comparison of results for the `satimage` data set.

	OVA	AVA	COM	DEN	SPA
Allwein et al.	72.9	40.9	40.4	39.7	47.2
Rifkin	39.0	39.3	38.5	39.3	38.8

Comparison of results for the `yeast` data set.

Fürnkrnaz: Round-Robin Classification

Using Ripper, a rule-based learner, as an underlying binary learner, Fürnrkanz showed experimentally that all-pairs substantially outperformed one-vs-all across a variety of data sets from UCI.

His experiments include the `satimage` and `yeast` data sets discussed above, with his best all-pairs system achieving accuracy rates of 10.4% and 41.8%, respectively. These results cannot be directly compared to my numbers (7.9% and 39.0%) because I cheated and because the `yeast` set-up was somewhat different, but see below.

Hsu and Lin: A Metastudy

Hsu and Lin empirically compared different methods of multiclass classification using SVMs as binary learners.

They tried five methods: OVA, AVA, DAG (similar to AVA, faster at testing time), the method of Crammer & Singer, and the method of Weston & Watkins. They used Gaussian kernels, tuned on validation sets, and reported all the numbers.

They conclude that "one-against-one and DAG methods are more suitable for practical use than the other methods." We take the numbers directly from their paper, and presented them in a format that suits us. . .

Hsu and Lin: Results

	Best	Worst	Diff	Size	Size * Diff
iris	97.333	96.667	.666	150	1.000
wine	99.438	98.876	.562	178	1.000
glass	73.832	71.028	2.804	214	6.000
vowel	99.053	98.485	.568	528	3.000
vehicle	87.470	86.052	1.418	746	10.58
segment	97.576	97.316	.260	2310	6.006
dna	95.869	95.447	.422	1186	5.005
satimage	92.35	91.3	1.050	2000	20.1
letter	97.98	97.68	.300	5000	15.0
shuttle	99.938	99.910	.028	14500	4.06

A view of the multiclass results of Hsu and Lin for RBF kernels.

New Experiments

With Aldebaro Klautau of UCSD, I performed a set of experiments on data from the UCI data set. There were three main things we wanted to explore, in a well-controlled, (hopefully) reproducible setting:

- Revisiting the data sets Fürnkranz used, but using well-tuned SVMs as the base learners.

- Comparing the five different matrices of Allwein et al., across a range of data sets.

- Comparing RLSC and SVM.

Protocol

Details of the protocol are found in the papers. All experiments were done with a Gaussian kernel. For a given experiments, the same parameters (σ and C (or λ)) were used for all machines. All parameters were found by cross-validation.

We focussed specifically on data sets on which Fürnkranz had found a significant difference between OVA and AVA. But what is significance?

McNemar's Test, I

Fürnkranz used *McNemar's test* to assess the significance of the difference in performance of two classifiers.

We compute the number of times (over the test set) each of these events occur:

- both classifiers were correct on (CC)
- both classifiers were incorrect (II)
- A was correct, B incorrect (CI)
- B was correct, A incorrect (IC)

McNemar's Test, II

McNemar's Test uses the observation that, if the classifiers have equal error rates, then CI and IC should be equally frequent. This test is good because it requires very few assumptions (just that the observations are paired). It's bad because it ignores the number of examples on which the two systems agree, and (directly related) it does not provide a confidence interval.

We therefore decided to introduce...

A Bootstrap Test For Comparing Classifiers

We calculate the empirical probabilities of the four events CC, CI, IC, and II. Then, a large number (in our experiments, 10,000) of times, we generate *bootstrap samples* containing ℓ "data points", each of which is simply an occurrence of CC, CI, IC, or II with the appropriate probability. We then calculate confidence intervals (in our experiments, 90%) on the difference in performance (CI - IC).

Other, better approaches may be possible.

Data Sets Used

Name	train	test	class	# att / # nom	average / min / max # examples per class	baseline error (%)
soybean-large	307	376	19	35 / 35	16.2 / 1 / 40	87.2
letter	16000	4000	26	16 / 0	615.4 / 576 / 648	96.4
satimage	4435	2000	6	36 / 0	739.2 / 409 / 1059	77.5
abalone	3133	1044	29	8 / 1	108.0 / 0 / 522	84.0
optdigits	3823	1797	10	64 / 0	382.3 / 376 / 389	89.9
glass	214	-	7	9 / 0	30.6 / 0 / 76	64.5
car	1728	-	4	6 / 6	432.0 / 65 / 1210	30.0
spectrometer	531	-	48	101 / 0	11.1 / 1 / 55	89.6
yeast	1484	-	10	8 / 0	148.4 / 5 / 463	68.8
page-blocks	5473	-	5	10 / 0	1094.6 / 28 / 4913	10.2

Reproducing Fürnkranz's Experiments

Data Set	Current Experiments	Furnkranz's paper
soybean-large	13.3	6.30
letter	7.7	7.85
satimage	12.2	11.15
abalone	74.1	74.34
optdigits	7.5	3.74
glass	26.2	25.70
car	2.8	2.26
spectrometer	51.2	53.11
yeast	41.6	41.78
page-blocks	2.6	2.76

Code Matrix Sizes

Name	OVA	AVA	COM	DEN	SPA
soybean-large	19	171	262143	43	64
letter	26	325	33554431	48	71
satimage	6	15	31	26	39
abalone	29	406	268435455	49	73
optdigits	10	45	511	34	50
glass	6	15	31	26	39
car	4	6	7	20	30
spectrometer	48	1128	1.407375e+014	56	84
yeast	10	45	511	34	50
page-blocks	5	10	15	24	35

Results: SVM AVA vs. OVA

Data Set	AVA	OVA	DIFF	AGREE	BOOTSTRAP
soybean-large	6.38	5.85	0.530	0.971	[-0.008, 0.019]
letter	3.85	2.75	1.09	0.978	[0.008, 0.015]
satimage	8.15	7.80	0.350	0.984	[-5E-4, 0.008]
abalone	72.32	79.69	-7.37	0.347	[-0.102, -0.047]
optdigits	3.78	2.73	1.05	0.982	[0.006, 0.016]
glass	30.37	30.84	-.470	0.818	[-0.047, 0.037]
car	0.41	1.50	-1.09	0.987	[-0.016, -0.006]
spectrometer	42.75	53.67	-10.920	0.635	[-0.143, -0.075]
yeast	41.04	40.30	0.740	0.855	[-0.006, 0.021]
page-blocks	3.38	3.40	-.020	0.991	[-0.002, 0.002]

Results: SVM DENSE vs. OVA

Data Set	DEN	OVA	DIFF	AGREE	BOOTSTRAP
soybean-large	5.58	5.85	-0.270	0.963	[-0.019, 0.013]
letter	2.95	2.75	0.200	0.994	[5E-4, 0.004]
satimage	7.65	7.80	-0.150	0.985	[-0.006, 0.003]
abalone	73.18	79.69	-6.51	0.393	[-0.092, -0.039]
optdigits	2.61	2.73	-0.12	0.993	[-0.004, 0.002]
glass	29.44	30.84	-1.40	0.911	[-0.042, 0.014]
car	-	1.50	-	-	-
spectrometer	54.43	53.67	-0.760	0.866	[-0.011, 0.026]
yeast	40.30	40.30	0.00	0.900	[-0.011, 0.011]
page-blocks	-	3.40	-	-	-

Results: SVM SPARSE vs. OVA

Data Set	SPA	OVA	DIFF	AGREE	BOOTSTRAP
soybean-large	6.12	5.85	0.270	0.968	[-0.011, 0.016]
letter	3.55	2.75	0.800	0.980	[0.005, 0.011]
satimage	8.85	7.80	1.05	0.958	[0.003, 0.018]
abalone	75.67	79.69	-4.02	0.352	[-0.067, -0.014]
optdigits	3.01	2.73	0.280	0.984	[-0.002, 0.008]
glass	28.97	30.84	-1.87	0.738	[-0.070, 0.033]
car	0.81	1.50	-0.69	0.988	[-0.011, -0.003]
spectrometer	52.73	53.67	-0.940	0.744	[-0.038, 0.019]
yeast	40.16	40.30	-0.140	0.855	[-0.015, 0.013]
page-blocks	3.84	3.40	0.440	0.979	[0.001, 0.007]

Results: SVM COMPLETE vs. OVA

Data Set	COM	OVA	DIFF	AGREE	BOOTSTRAP
soybean-large	-	5.85	-	-	-
letter	-	2.75	-	-	-
satimage	7.80	7.80	0.00	0.999	[-1E-3, 1E-3]
abalone	-	79.69	-	-	-
optdigits	2.67	2.73	-0.060	0.996	[-0.003, 0.002]
glass	29.44	30.84	-1.340	0.911	[-0.042, 0.014]
car	1.68	1.50	-0.180	0.998	[5.79E-4, 0.003]
spectrometer	-	53.67	-	-	-
yeast	38.61	40.30	-1.690	0.906	[-0.028, -0.005]
page-blocks	3.49	3.40	-0.090	0.983	[-0.002, 0.004]

Results: FÜRNRKANZ vs. SVM OVA

Data Set	FUR	OVA	DIFF	AGREE	BOOTSTRAP
soybean-large	13.3	5.85	7.45	0.891	[.056, .109]
letter	7.7	2.75	4.95	0.922	[.043, .057]
satimage	12.2	7.80	4.40	0.906	[.0345, .055]
abalone	74.1	79.69	-5.59	0.335	[-.083, -0.029]
optdigits	7.5	2.73	4.77	0.920	[0.035, 0.056]
glass	26.2	30.84	-4.64	0.734	[-0.098, 0.005]
car	2.8	1.50	1.3	0.969	[0.006, 0.020]
spectrometer	51.2	53.67	-2.47	0.488	[-0.060, 0.017]
yeast	41.6	40.3	1.29	0.765	[-0.005, -0.032]
page-blocks	2.6	3.40	-0.80	0.978	[-0.012, -0.005]

Results: SVM OVA vs. RLSC OVA

Data Set	RLSC	SVM	DIFF	AGREE	BOOTSTRAP
soybean-large	6.12	5.85	0.270	0.984	[-0.008, 0.013]
letter	-	2.75	-	-	-
satimage	7.9	7.80	0.010	0.979	[-0.004, 0.006]
abalone	72.7	79.69	-7.000	0.284	[-0.099, -0.041]
optdigits	2.5	2.73	-0.230	0.980	[-0.007, 0.003]
glass	31.3	30.84	0.460	0.808	[-0.037, 0.047]
car	2.9	1.50	1.40	0.980	[0.009, 0.020]
spectrometer	52.3	53.67	-1.370	0.821	[-0.036, 0.009]
yeast	40.0	40.30	-0.300	0.872	[-0.016, 0.011]
page-blocks	3.25	3.40	-0.150	0.983	[-0.004, 0.001]

Results: SVM AVA vs. RLSC AVA

Data Set	RLSC	SVM	DIFF	AGREE	BOOTSTRAP
soybean-large	8.2	6.38	1.820	0.941	[0.000, 0.037]
letter	-	3.85	-	-	-
satimage	7.4	8.15	-.750	0.974	[-0.013, -0.001]
abalone	73.66	72.32	1.340	0.560	[-0.009, 0.034]
optdigits	3.0	3.78	-.780	0.974	[-0.013, -0.002]
glass	29.4	30.37	-0.970	0.864	[-0.047, 0.028]
car	2.3	0.41	1.89	0.980	[0.013, 0.024]
spectrometer	49.1	42.75	6.350	0.738	[0.036, 0.092]
yeast	40.0	41.04	-1.040	0.838	[-0.025, 0.005]
page-blocks	3.4	3.38	0.020	0.981	[-0.003, 0.003]

Results: SVM DENSE vs. RLSC DENSE

Data Set	RLSC	SVM	DIFF	AGREE	BOOTSTRAP
soybean-large	8.0	5.58	2.41	0.971	[0.011, 0.040]
letter	8.0	7.65	0.350	0.976	[-0.002, 0.009]
abalone	72.8	73.18	-0.380	0.663	[-0.025, 0.017]
optdigits	2.5	2.61	-0.110	0.982	[-0.006, 0.003]
glass	29.9	29.44	-.460	0.864	[-0.037, 0.042]
car	-	-	-	-	-
spectrometer	52.9	54.43	-1.530	0.825	[-0.038, 0.008]
yeast	40.0	40.30	-.300	0.888	[-0.016, 0.009]
page-blocks	-	-	-	-	-

Results: SVM SPARSE vs. RLSC SPARSE

Data Set	RLSC	SVM	DIFF	AGREE	BOOTSTRAP
soybean-large	7.4	6.12	1.280	0.973	[0.000, 0.027]
letter	-	3.55	-	-	-
satimage	8.4	8.85	-0.450	0.958	[-0.011, 0.003]
abalone	73.3	75.67	-2.370	0.621	[-0.043, -0.005]
optdigits	3.6	3.01	0.590	0.977	[0.001, 0.011]
glass	29.4	28.97	-0.430	0.841	[-0.037, 0.047]
car	4.3	0.81	3.490	0.963	[0.028, 0.043]
spectrometer	52.5	52.73	-0.230	0.827	[-0.024, 0.021]
yeast	40.9	40.16	0.740	0.877	[-0.005, 0.020]
page-blocks	3.5	3.84	-0.340	0.980	[-0.006, 1.83E-4]

Towards a Theory, I

Recall that to solve an RLSC problem, we solve a linear system of the form

$$(K + \lambda \ell I)\mathbf{c} = \mathbf{y}.$$

Because this is a *linear* system, the vector $\mathbf{c}$ is a linear function of the right hand side $\mathbf{y}$. Define the vector $\mathbf{y^i}$ as

$$y_j^i = \begin{cases} 1 & \text{if } y_j = i \\ 0 & \text{otherwise} \end{cases}$$

Towards a Theory, II

Now, suppose that we solve N RLSC problems of the form

$$(K + \lambda \ell I)\mathbf{c^i} = \mathbf{y^i},$$

and denote the associated functions $f^{c_1}, \ldots, f^{c_N}$. Now, for *any* possible right hand side $\mathbf{y}^*$ for which the y_i and y_j are equal, whenever $\mathbf{x_i}$ and $\mathbf{x_j}$ are in the same class, we can calculate the associated $\mathbf{c}$ vector from the $\mathbf{c^i}$ *without solving a new RLSC system*. In particular, if we let m_i ($i \in \{1, \ldots, N\}$) be the y value for points in class i, then the associated solution vector $\mathbf{c}$ is given by

$$\mathbf{c} = \sum_{i=1}^{N} c^i m_i.$$

Towards a Theory, III

For any code matrix M not containing any zeros, we can compute the output of the coding system using only the entries of the coding matrix and the outputs of the underlying one-vs-all classifiers. In particular, we do not need to actually train the classifiers associated with the coding matrix. We can simply use the appropriate linear combination of the "underlying" one-vs-all classifiers. For any $r \in \{1, \ldots, N\}$, it can be shown that

$$\begin{aligned} & \sum_{i=1}^{F} L(M_{ri} f_i(\mathbf{x})) \\ = & \sum_{\substack{j=1 \\ j \neq r}}^{N} D_{rj} f^j(\mathbf{x}), \end{aligned} \tag{1}$$

where $C_{rj} \equiv \sum_{i=1}^{F} M_{ri} M_{ji}$ and $D_{rj} \equiv (F - C_{rj})$.

Towards a Theory, IV

Noting that D_{rj} is guaranteed to be positive under the basic assumption that no two rows of the coding matrix are identical, we see that

$$\begin{aligned} f(\mathbf{x}) &= \arg \min_{r \in \{1,\ldots,N\}} \sum_{i=1}^{F} L(M_{ri} f_i(\mathbf{x})) \\ &= \arg \min_{r \in \{1,\ldots,N\}} \sum_{\substack{j=1 \\ j \neq r}}^{N} D_{rj} f^j(\mathbf{x}). \end{aligned}$$

Towards a Theory: Class Symmetry I

We define a coding matrix to be *class-symmetric* if D_{ij} (and therefore C_{ij} as well) is independent of i and j (assuming $i \neq j$). Note that a sufficient (but not necessary) condition for a coding-matrix to class symmetric is if, whenever it contains a column containing k 1's and $n-k$ -1's, all $\frac{N!}{k!(N-k)!}$ such columns are included. The OVA and COMPLETE schemes are class-symmetric, while the DENSE scheme is in general not (the AVA and SPARSE schemes include zeros in the coding matrix, and are not addressed by this analysis).

Towards a Theory: Class Symmetry II

For class-symmetric schemes, D_{rj} is independent of the choice of r and j, and can be denoted as D^*. For these schemes,

$$\begin{aligned}
f(\mathbf{x}) &= \arg\min_{r \in 1,\ldots,N} D^* \sum_{\substack{j=1 \\ j \neq r}}^{N} f^j(\mathbf{x}) \\
&= \arg\min_{r \in 1,\ldots,N} \sum_{\substack{j=1 \\ j \neq r}}^{N} f^j(\mathbf{x}) \\
&= \arg\max_{r \in 1,\ldots,N} f^r(\mathbf{x}),
\end{aligned}$$

When RLSC is used as the underlying binary learner for class-symmetric coding matrices containing no zeros, the predictions generated are identical to those of the one-vs-all scheme.