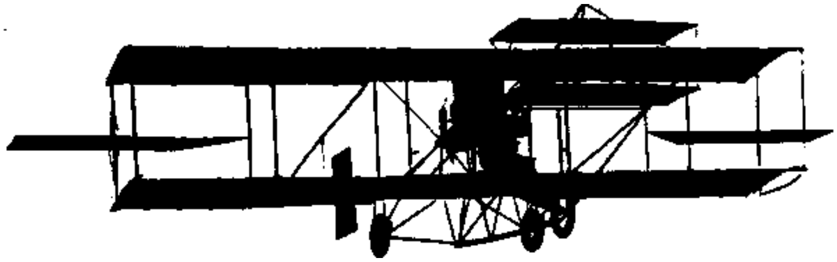
The ______________________________



__ INSTITUTION

periodical Newsletter

December 27, 2023

`https://peteysoft.github.io`

Table of contents

Directions and recap

When I look back, it feels like I've been spending most of my time in bed, but when I actually take stock of my achievements, the last few years have been very productive. I published three papers in peer-reviewed journals along with two in-depth summary papers on the arXiv preprint database. This is in addition to two conference presentations, thousands of lines of computer code, including several new scientific software packages, and innumerable popular science articles on social media.

Research proceeded on two fronts: machine learning, mainly statistical classification, following on my work using it for remote sensing retrievals, and tracer transport, especially the principal component proxy tracer dynamical interpolation technique. Much of this was simply cleaning up existing work and tying up loose ends.

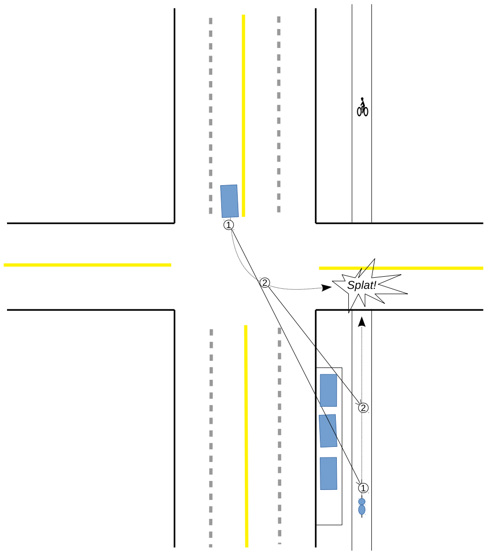
I like to think that my summary of multi-class classification, entitled, "Solving for multi-class: a survey and synthesis" (arXiv: 1809.05929) is both the first and last word on the matter. Along with the article, there is also software that implements the ideas in the paper (`https://github.com/peteysoft/libmsci`). This article has collected six citations already. This newsletter contains a summary as it was presented to the Machine Learning and Artificial Intelligence Ottawa meetup group on June 25, 2019.

Next, I have finally published my work on dynamical tracer interpolation in the peer-reviewed literature (*Environmental Fluid Mechanics* **18** (6): 1533). To help get there, I compiled a lengthy summary of mathematical techniques related to modelling tracer transport using matrices and linear algebra, entitled, "Matrix Analysis of Tracer Transport" (arXiv: 1506:06984). A summary of this latter work is also contained in this newsletter. It was originally presented to the 2019 International conference on Water, Informatics, Sustainability and the Envirionment (iWISE) at Carleton University. Applications include monitoring ozone and other long-lived atmospheric tracers, as well as deriving sources and sinks for these tracers. Maps of carbon dioxide sources and sinks would be of particular interest in today's, increasingly industrialized world.

The focus of new work has taken a different direction. First, I have started compiling a work that generalizes the tracer transport equations to classical systems modelled as evolving probability distributions. This has applications in non-equilibrium statistical mechanics and the understanding of entropy.

Second, I've become interested in the difference between segregated cycle lanes and the road. This is very much in line with the whole spirit of my work. It's gratifying to research something highly theoretical and abstract. One feels inside of a rarified atmosphere, exploring the corners of some Platonic realm. And of course we can't forget the prestige often associated with such work.

But it is the more mundane aspects of our world that often need the most attention, as too many researchers fail to apply sufficient rigour to everyday problems. Often, problems in the social sciences are considerably more difficult than those in the "hard" sciences because they are so complex. The problems with segregated bicycle lanes, for instance, have been understood for decades, and can be illustrated by simple diagrams like this one:



Yet many modern traffic engineers insist that this is the only means of making cycling a safe and viable tranportation means for everyone. How can we resolve the dispute between so-called "vehicular cyclists" and those advocating for Dutch-style segregated lanes? There is plenty of work, of course, on the safety of cycle lanes relative to the road. Most of this, however, compares them as a statistical aggregate, with little attempt at removing or understanding confounding factors. We can compare the safety of cyclists in the U.S. versus the Netherlands, for instance, and infer that the larger number and type of cycle lanes in the Netherlands results in their superior safety record. There are many other differences between the U.S. and the Netherlands other than simply "cycling infrastrucutre", however. Most who have travelled to Europe likely noted the superiority of European drivers over their American counterparts. What if this is the actual reason for the difference in cyclist accident rates? Certainly it could be a confounding factor, along with many other possible ones.

My interest is in comparing the roads and cycle lanes directly, while controlling (ideally) for cyclist skill and power. My preliminary work involved simply riding the bike lane, then the road, and then comparing them on the basis of four factors: length, speed, surface quality, and number of traffic conflicts. The latter would be defined as any time I must take evasive action to avoid a collision.

Since I've ridden these lanes and know what they're like, I suspect I already know the results. In my initial trials (three comparisons, so far), I've found the lanes to be worse in every way than the road: longer, slower, rougher, and with more traffic conflicts. This seems less about science, however, than polemics. It has been found that cyclists in bike lanes tend to ride more slowly than those on the road. Stopping distance is proportional to the square of the speed, thereby directly affecting safety. So at the very least, we need to get a handle on how speed affects the outcome. Another important variable is rider experience, as noted above.

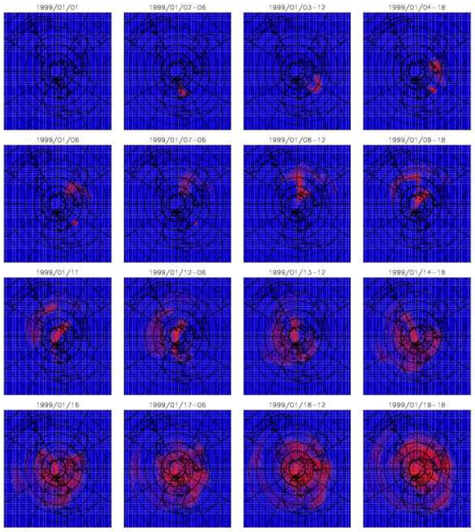
My new plan, therefore, is to ride each lane multiple times while maintaining (in as much as possible) a constant power output. Outputs of 50, 100, 150, and (if I can manage it) 200 Watts could be tried. Not only would this help understand how speed affects the results, but also get a direct measure of the efficiency of each route as a measure of distance per unit energy. A potentially less obvious benefit would be mitigating any implicit bias by the rider (me). A better experimental design would be randomized and double blind, but I can certainly gather some preliminary results. The real measure of success in the experimental sciences is replicability, not necessarily rigour.

If you are interested in contributing to any of these projects, whether money, time, or resources, please drop me a line at *peteymills@hotmail.com*. Power meters are expensive, and newer versions require an expensive portable phone to make them work, so this would be a welcome donation. Also, if you are free to ride and record your own comparison of road versus bike lane (especially segregated), that would be an excellent contribution.

Peter Mills

Linear maps for modelling tracer transport

a mathematical overview



International Conference on Water, Informatics, Sustainabiliy, and Environment (iWISE)
Carleton University, Ottawa, August 9, 2019.

Basic Idea

Advection-diffusion equation:

$$\frac{\nabla q}{\nabla t} = \left[-\vec{v} \cdot \nabla + \nabla \cdot D \cdot \nabla \right] q$$

where:

- $q = q(\vec{x}, t)$ is the tracer concentration
- t is time
- $\vec{v} = \vec{(}\vec{x}, t)$ is velocity
- $\vec{x}$ is spatial position
- d is the diffusion coefficient

Represent bracketed section as a linear map:

$$\frac{\partial q}{\partial t} = A \cdot q \tag{1}$$

Linear maps

Properties of linearity:

$$\begin{aligned} A \cdot (cq) &= c(A \cdot q) \\ A \cdot (q + p) &= A \cdot q + A \cdot p \end{aligned}$$

where c is a scalar constant.

Function map (theoretical):

$$A(\vec{x}, \vec{x}') \cdot q(\vec{x}') = \int A(\vec{x}, \vec{x}' q(\vec{x}') \mathrm{d}\vec{x}'$$

Matrix(actual):

$$A \cdot \vec{q} = \left\{ \sum_j a_{ij} q_j \right\}$$

(Analysis is often identical.)

Solution

$$\begin{aligned} \frac{\partial R(t_0, \Delta t)}{\partial (\Delta t)} &= A(t_0 + \Delta t) \cdot R(t_0, \Delta t) \qquad (2) \\ R(t_0, 0) &= I \end{aligned}$$

Where:

- $R = R(\vec{x}, \vec{x}_0, t_0, \Delta t)$ is a linear map
- I is the identity map: $I \cdot q = q$
- $t_n = t_0 + \sum_{i=1}^{n} \Delta t_i$

There are two parameters for time so that R may be decomposed in terms of itself:

$$\begin{aligned} &R(t_0, t_n - t_0) = \\ &\quad R(t_{n-1}, \Delta t_n) \cdot R(t_{n-2}, \Delta t_{n-1}) \cdot \ldots \\ &\quad \cdot R(t_2, \Delta t_3) \cdot R(t_1, \Delta t_2) \cdot R(t_0, \Delta t_1) \end{aligned}$$

The final tracer field is determined by multiplying R with the initial tracer field:

$$q = R(t_0, t - t_0) \cdot q_0$$

R approaches a steady-state solution as the time-step becomes small:

$$\lim_{\Delta t \to 0} \{R(t, \Delta t) = \exp\left[\Delta t A(t)\right]\}$$

Eigenvalue decomposition:

$$\begin{aligned} A(t) &= V \cdot \Lambda \cdot V^{-1} \\ \lim_{\Delta t \to 0} \{R(t, \Delta t) &= V^{-1} \cdot \exp\left[\Delta t \Lambda\right] \cdot V\} \end{aligned}$$

Where exp is the vector generalization of the exponential function.

Operators to linear maps

Consider the identity operator:

$$q(\vec{x}) = \int \delta(\vec{x} - \vec{x}')q(\vec{x}')\mathrm{d}\vec{x}'$$

Take the derivative:

$$\nabla_{\vec{x}} q = \int \nabla_{\vec{x}} \delta(\vec{x} - \vec{x}')\mathrm{d}\vec{x}'$$

$$\frac{\partial q}{\partial x_i} = \int \delta'(x_i - x_i') \prod_{j \neq i} \delta(x_j - x_j')q(x_i')\mathrm{d}\vec{x}$$

Where:

- $\delta(\vec{x})$ is the vector generalization of the Dirac delta function:

$$\delta(\vec{x}) = \prod_i \delta(x_i)$$

- δ' is the generalized derivative of the Dirac delta function:

$$\int \delta'(x - x')q(x')\mathrm{d}x' = q(x')$$

Adding sources and sinks

$$\begin{aligned} q(t_n) \quad &\approx \quad [\ldots[[[q_0 \cdot q_0 \cdot R(t_0, \Delta t) + \sigma(t_1)] \cdot \\ &\qquad R(t_1, \Delta t) + \sigma(t_2)] \cdot \\ &\qquad R(t_2, \Delta t) + \sigma(t_3)] \cdot \ldots] \\ &= \quad R(t_0, t_n - t_0) \cdot q_0 + \\ &\qquad \sum_{i=1}^{n} R(t_i, t_n - t_i) \cdot \sigma(t_i) \end{aligned} \tag{3}$$

Where σ is the integrated source term.

Applications

Genesis

- Studied noisy chaos for undergrad. thesis.
- Represent classical systems as probability distributions (mathematics is equivalent).
- Principal component proxy tracer dynamical tracer interpolation method.
- Reviewers were interested in error analysis and parameter selection.

(preprint: Peter Mills (2014) "Matrix Analysis of Tracer Transport" arXiv: 1506.06984)

PC proxy Decompose R using SVD:

$$R(t_0, \Delta t) = U \cdot S \cdot V^T$$

Where:

- U is an orthogonal matrix of left singular vectors
- S is a diagonal matrix of singular values
- V is an orthogonal matrix of right singular vectors

Keep k largest singular values and correlate singular vectors with measurements:

$$\min_{\vec{c}} \sum_i \left\{ \sum_{j=1}^{k} c_j \vec{w}_i R(t_0, t^{(i)} - t_0) \vec{v}^{(j)} - m_i \right\}^2$$

Peter Mills (2018 "PC proxy: a method for dynamical tracer interpolation." *Environmental Fluid Mechanics* **18** (6): 1533-1558. Springer.

Retrieving sources and sinks Retroplumes

Adjoint equation:

$$\frac{\partial R^{-1}(t_0, \Delta t)}{\partial(\Delta t)} = R^{-1}(t_0, \Delta t) \cdot A(t_0 + \Delta t)$$

(Equation (2) with RHS transposed and negated.) Where R^{-1} is the inverse mapping. Map detector directly onto sources: $q_0 = R^{-1} \cdot q$

Numerical inversion

Recalling Equation (3), discretized and interpolated:

$$C \cdot \vec{\sigma} = \vec{\mu}$$

Where:

- C is a matrix of coefficients
- $\vec{\sigma}$ are the sources
- $\vec{\mu}$ are the measurements

Likely not well posed: will need to be constrained and regularized.

Papers that employ some or most of these ideas

- Houweling, S. et al. (1999) "Inverse modelling of methane sources and sinks using the adjoint of a global transport model." *J. of Geophysical Research* **104** (D21): 26137.
- Issartel, J.-P. (2003) "Rebuilding sources of linear tracers after atmospheric concentration measurements." *Atmospheric Chemistry and Physics* **3** (6): 2125.
- Bocquet, M. (2005) "Reconstruction of an atmospheric tracer source using the principle of maximum entropy. I: Theory." *QJRMS* **121** (610): 2191.
- Lin, J.C. et al. (2003) "A near field tool for simulating the upstream influence of atmospheric observations: The STILT model." *J. of Geophysical Research* **108** (D16): 4493.

Software

- Ctraj: semi-Lagrangian tracer simulation that outputs sparse matrices
- sparse_calc: interactive calculator for sparse matrices

`https://github.com/peteysoft/libmsci`

Thanks!

Thanks to W.A. Eldin for inviting me...

Solving for multi-class *A survey and synthesis*

Peter Mills

Webpage: `https://peteysoft.github.io`
Software: `https://github.com/peteysoft/libmsci`

Email: *peteymills@hotmail.com*

This paper was presented as a poster to the Machine Learning and Artificial Intelligence Ottawa Meetup group on Wednesday June 26, 2019 at 7 Bayview Rd. It gives a short summary of different methods of generalizing binary classifiers to multi-class and describes a unifying framework for representing them. The idea is to construct customized solutions for different problems. The only problem that seemed to benefit from a customized solution, however, was a remote sensing dataset in which the classes were discretized continuum values.

It remains to be seen whether using statistical classification in this way for non-linear regression is competitive with a neural network, for instance. To this end, I have designed a "lightweight" neural network library. Sitting at roughly 6000 lines of code, it piggy-backs upon the GNU Scientific Library (GSL) for the optimization methods. The plan is to use the same neural network for statistical classifications as for nonlinear regression.

Abstract

Many of the best statistical classification algorithms are binary classifiers that can only distinguish between one of two classes. The number of possible ways of generalizing binary classification to multi-class increases exponentially with the number of classes. There is some indication that the best method will depend on the dataset. Therefore we are particularly interested in data-driven solution design, whether based on prior considerations or on empirical examination of the data. Here we demonstrate how a recursive control language can be used to describe a multitude of different partitioning strategies in multi-class classification, including those in most common use. We use it both to manually construct new partitioning configurations as well as to examine those that have been automatically designed.

Eight different strategies were tested on eight different datasets using a support vector machine (SVM) as the base binary classifier. Numerical results suggest that a one-size-fits-all solution consisting of one-versus-one is appropriate for most datasets. Three datasets showed better accuracy using different methods. The best solution for the most improved dataset exploited a property of the data to produce an uncertainty coefficient 36% higher (0.016 absolute gain) than one-vs.-one. For the same dataset, an adaptive solution that empirically examined the data was also more accurate than one-vs.-one while being faster.

Full paper: `https://arxiv.org/abs/1809.05929`

Outline of the problem

Many of the best statistical classifiers are *binary classifiers*, that is they can only distinguish between two different classes, e.g.:

- linear perceptrons
- logistic regression (Michie et al., 1994)
- support vector machines (SVM) (Müller et al., 2001)
- piecewise linear classiifers (Mills, 2018)

Two problems:

1. How do we partition the class labels so as to train the binary classifiers?
2. Given the estimated conditional probabilities of the binary classifiers:

$$\{P_i(c|\vec{x})\}$$

we want to find the conditional probabilities of the multi-class problem:

$$P(j|\vec{x}),\ j = [1..n_c]$$

Why use probabilities?

1. Provide useful extra information: quantify the accuracy of an estimate.

2. Relation between binary and multi-class probabilities is unique and derives rigorously from probability theory.

3. Binary classifiers that return a continuous decision function can be recalibrated to an approximate probability.

Error correcting codes

Use a *coding matrix*, A, to relate binary class probabilities to multi-class probabilities: (Dietterich and Bakiri, 1995)

$$r_i = \frac{\sum_j a_{ij} p_j}{\sum_j |a_{ij} p_j}$$

- $r_i = P_i(+1|\vec x) - P_i(-1|\vec x)$
- $p_j = P(j|\vec x)$

Rearrange to form linear system:

$$\begin{aligned} Q\vec p &= \vec r \\ q_{ij} &= (1 - |a_{ij}|)\, r_i \end{aligned}$$

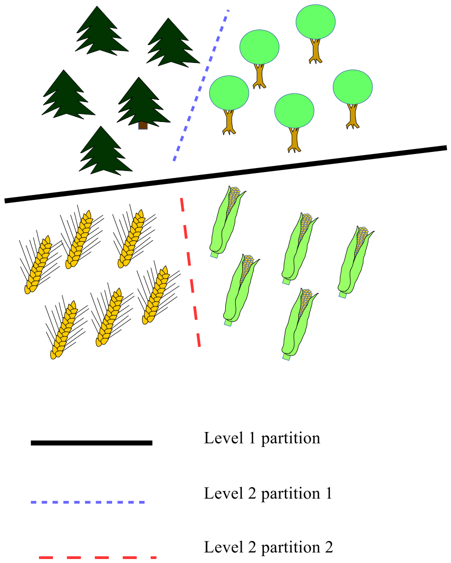
- Q reduces to A if A contains no zeros.
- A is transposed relative to normal convention: more natural when dealing with probabilities.
- All the above equations derive rigorously from probability theory.

Example coding matrix, one-vs.-one:

$$A = \begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Decision trees

Hierarchically subdivide the classes (Lee and Oh, 2003), such as in this hypothetical surface-classification probelm:



Only winning probability is returned as the product of all the probabilities returned at each level.

Empirical decision trees

- Use inter-class distance to build dendrogram (Benabdeslem and Bennani, 2006).
- Hausdorff distance (Ott, 1993):

$$\begin{aligned} & D_{Hij} = \\ & \quad \max\{\min_k |\vec x_k - \vec x_l|, \min_l |\vec x_k - \vec x_l|; \\ & \qquad y_k = i;\ y_l = j\} \end{aligned}$$

Unifying framework

Recursive control language:

```
<br>          ::=   <mod> “{” <br-list> “}” |
                    <CLS>
<mod>         ::=   <BIN> | <part-list>
<br-list>     ::=   <br> | <br-list> <br>
<part-list>   ::=   <part> | <part-list> <part>  .
<part>        ::=   <BIN> <cls-list> “ / ”
                    <cls-list> “;”
<cls-list>    ::=   <cls-list> <CLS> |
                    <CLS>
```

- <CLS> is a class value between 0 and $n_c - 1$.
- <BIN> is a binary classifier.

Example, one-vs.-one:

```
model01 0 / 1;
model02 0 / 2;
model03 0 / 3;
model12 1 / 2;
model13 1 / 3;
model23 2 / 3;
{0 1 2 3}
```

Example, the figure above:

```
TreeVsField {
  EvergreenVsDeciduous {0 1}
  CornVsWheat {2 3}
}
```

The two methods, error-correcting-codes and decision trees, can be combined. This is actually used in Zhou et al. (2019).

Numerical trials

Six configurations:

- 1-vs.-the-rest
- 1-vs.-1
- Orthogonal coding matrix (Mills, 2019)
- “Adjacent” partitioning*
- Arbitrary tree
- Empirically-designed tree

* Adjacent partitioning for seven classes:

```
shuttle_adj-00 0 / 1 2 3 4 5 6;
shuttle_adj-00 0 1 / 2 3 4 5 6;
shuttle_adj-00 0 1 2 / 3 4 5 6;
shuttle_adj-00 0 1 2 3 / 4 5 6;
shuttle_adj-00 0 1 2 3 4 / 5 6;
shuttle_adj-00 0 1 2 3 4 5 / 6;
{0 1 2 3 4 5 6}
```

Example of empirically-designed tree for **sat** dataset:

```
sat_emp {
  sat_emp.00 {
    sat_emp.00.00 {
      sat_emp.00.00.00 {
        sat_emp.00.00.00.00 {
          VERY DAMP GREY SOIL
          DAMP GREY SOIL
        }
        RED SOIL
      }
      GREY SOIL
    }
    STUBBLE
  }
  COTTON CROP
}
```

Results

Uncertainty coefficient (Mills, 2018):

config.	method	letter	pendigits	usps	segment
1 vs. 1	inv.	**0.940 ± 0.002**	**0.986 ± 0.003**	**0.931 ± 0.009**	0.922 ± 0.010
1 vs. rest	lsq.	0.932 ± 0.003	0.982 ± 0.004	0.928 ± 0.007	0.921 ± 0.010
ortho.	iter.	0.922 ± 0.003[*]	0.982 ± 0.003	0.927 ± 0.008	**0.923 ± 0.010**
adj.	lsq.	0.886 ± 0.005	0.971 ± 0.004	0.906 ± 0.008	0.913 ± 0.010
hier	rec.	0.880 ± 0.004	0.972 ± 0.004	0.910 ± 0.007	0.909 ± 0.008
	lsq.	0.888 ± 0.004	0.973 ± 0.004	0.912 ± 0.007	0.909 ± 0.008
emp.	rec.	0.905 ± 0.003	0.979 ± 0.003	0.917 ± 0.008	0.903 ± 0.006
	lsq.	0.910 ± 0.003	0.979 ± 0.004	0.920 ± 0.010	0.909 ± 0.007

[*] A random coding matrix was used since building an orthogonal matrix would take too long using current methods.

config.	method	sat	urban	shuttle	humidity
1 vs. 1	inv.	**0.800 ± 0.010**	0.729 ± 0.030	**0.982 ± 0.003**	0.432 ± 0.006
1 vs. rest	lsq.	0.799 ± 0.009	0.728 ± 0.030	0.979 ± 0.003	0.359 ± 0.007
ortho.	iter.	0.798 ± 0.010	0.730 ± 0.030	0.974 ± 0.002	0.403 ± 0.009
adj.	lsq.	0.792 ± 0.010	**0.735 ± 0.030**	0.970 ± 0.002	**0.448 ± 0.006**
hier	rec.	0.788 ± 0.010	0.724 ± 0.030	0.974 ± 0.003	0.435 ± 0.006
	lsq.	0.789 ± 0.010	0.727 ± 0.030	0.973 ± 0.002	0.433 ± 0.007
emp.	rec.	0.790 ± 0.009	0.702 ± 0.050	0.977 ± 0.004	0.440 ± 0.008
	lsq.	0.795 ± 0.010	0.714 ± 0.040	0.975 ± 0.003	0.437 ± 0.008

Brier score (Jolliffe and Stephenson, 2003):

config.	letter	pendigits	usps	segment
1 vs. 1	**0.0480 ± 0.0008**	**0.032 ± 0.002**	**0.066 ± 0.003**	0.090 ± 0.005
1 vs. rest	0.0519 ± 0.0007	0.035 ± 0.002	0.070 ± 0.003	0.092 ± 0.004
ortho.	0.0587 ± 0.0006[*]	0.037 ± 0.002	0.070 ± 0.003	**0.090 ± 0.006**
adj.	0.063 ± 0.001	0.042 ± 0.002	0.077 ± 0.003	0.093 ± 0.007
hier.	0.062 ± 0.001	0.040 ± 0.002	0.075 ± 0.003	0.095 ± 0.005
emp.	0.0553 ± 0.0008	0.035 ± 0.003	0.071 ± 0.004	0.094 ± 0.003

config.	sat	urban	shuttle	humidity
1 vs. 1	**0.145 ± 0.004**	0.170 ± 0.006	0.018 ± 0.001	**0.259 ± 0.001**
1 vs. rest	0.149 ± 0.003	0.171 ± 0.007	**0.012 ± 0.001**	0.2750 ± 0.0009
ortho.	0.149 ± 0.003	0.172 ± 0.005	0.022 ± 0.001	0.268 ± 0.002
adj.	0.152 ± 0.004	**0.167 ± 0.008**	0.0248 ± 0.0007	0.264 ± 0.002
hier.	0.150 ± 0.004	0.167 ± 0.007	0.023 ± 0.001	0.259 ± 0.001
emp.	0.149 ± 0.004	0.173 ± 0.008	0.022 ± 0.001	0.261 ± 0.002

Conclusions

- One-size-fits-all method adequate for most datasets.
- One-vs.-one most accurate.
- Decision tree fastest (see full paper) (Mills, 2018).
- Some datasets benefit from a customized approach.
- Because the **humidity** dataset is a discretized continuum problem there is an ordering to the classes, hence "adjacent" partitioning works best.
- Recursvie control language can help find best configuration.

References

Benabdeslem, K. and Bennani, Y. (2006). Dendrogram-based SVM for Multi-Class Classification. *Journal of Computing and Information Technology*, 14(4):283–289.

Dietterich, T. G. and Bakiri, G. (1995). Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research*, 2:263–286.

Jolliffe, I. T. and Stephenson, D. B. (2003). *Forecast Verification: A Practitioner's Guide in Atmospheric Science.* Wiley.

Lee, J.-S. and Oh, I.-S. (2003). Binary Classification Trees for Multi-class Classification Problems. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, volume 2, pages 770–774. IEEE Computer Society.

Michie, D., Spiegelhalter, D. J., and Tayler, C. C., editors (1994). *Machine Learning, Neural and Statistical Classification.* Ellis Horwood Series in Artificial Intelligence. Prentice Hall, Upper Saddle River, NJ. Available online at: `http://www.amsta.leeds.ac.uk/~charles/statlog/`.

Mills, P. (2018). Accelerating kernel classifiers through borders mapping. *Real-Time Image Processing.* doi:10.1007/s11554-018-0769-9.

Mills, P. (2019). Solving for multiclass using orthogonal coding matrices. *SN Applied Sciences*, 1(11):1451.

Müller, K.-R., Mika, S., Rätsch, G., Tsuda, K., and Schölkopf, B. (2001). An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201.

Ott, E. (1993). *Chaos in Dynamical Systems.* Cambridge University Press.

Zhou, J. T., Tsang, I. W., Ho, S.-S., and Mueller, K.-R. (2019). N-ary decomposition for multi-class classification. *Machine Learning.* doi:10.1007/s10994-019-05786-2.

Boolean Algebra in a Nutshell

This article was originally published on Medium on May 29, 2018.
https://medium.com/@peteymills/boolean-algebra-in-a-nutshell-accab8430266

Introduction

Table 1: List of operators used in Boolean algebra. All operators return a new binary proposition.

Name	Symbol	Example
AND	$\wedge$	$X \wedge Y$
OR	$\vee$	$X \vee Y$
NOT	$\neg$	$\neg X$
if-then	$\rightarrow$	$X \rightarrow Y$
if-and-only-if	$\iff$	$X \iff Y$
exclusive OR (XOR)	$\oplus$	$X \oplus Y$

The usual approach to teaching logic is to start with Aristotelian logic. In today's digital world, I think a more natural starting point for the topic is Boolean algebra. Not only does it flow more naturally into more modern logics such as propositional and first-order logic, it is also invaluable to both the computer programmer and computer engineer.

Let P be a proposition, for instance, "*Roses are red*," "*The sky is blue*," or "*All dogs are mammals.*" As such, P will take on a truth value. In Boolean logic, it may take on only one of two values: that is, it is either true or false, 0 or 1; $P \in \{0,\ 1\}$.

The purpose of Boolean algebra is to form conjunctive expressions with one or more of these propositions. On the basis of the truth values of the individual propositions, the truth of the entire expression may be evaluated.

Two of the most common conjunctions used in English to unite two or more statements (propositions) are AND and OR, for instance:

Roses are red AND *the sky is blue.*

These conjunctions are also used in Boolean logic as *operators* between two propositions, although OR is somewhat different than how it is used in English.

The two symbols for AND and OR are $\wedge$ and $\vee$, respectively. Suppose we have two propositions, P and Q. We can use these operators to unite them, thereby turning them into a new proposition:

$$P \wedge Q$$

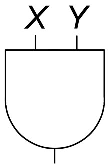
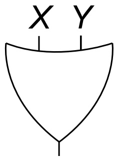
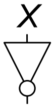
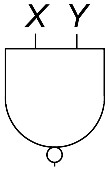
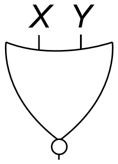
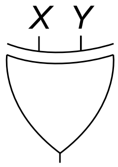
which is read, "P AND Q". Another useful operator is the NOT or negation operator, $\neg$. This is a *unary* rather than a *binary* operator, hence operates on only one argument:

$$\neg P$$

An interesting feature of Boolean logic, is that unlike in older forms of logic, such as Aristotelian logic, the entire content of an elementary proposition, such as P, above, is abstracted away, leaving only the binary truth value.

Truth tables

Table 2: Table of gates and equivalent Boolean expressions.

Name	**Gate**	**Expression**
AND		$X \wedge Y$
OR		$X \vee Y$
NOT		$\neg X$
NAND		$\neg(X \wedge Y)$
NOR		$\neg(X \vee Y)$
XOR		$X \oplus Y$

As mentioned in the previous section, combining one or more propositions with an operator creates a new proposition. How do we determine the truth value of this new proposition? One way is using a *truth table.* A truth table tabulates the truth value of the whole expression for every possible truth value of each of the consituent propositions.

Here is the *truth table* for the AND operator:

X	Y	$X \wedge Y$
0	0	0
0	1	0
1	0	0
1	1	1

Note that the logical OR is different from the English “or”. In English speech, “or” is the equivalent to the exclusive OR (XOR) which will be discussed in next section.

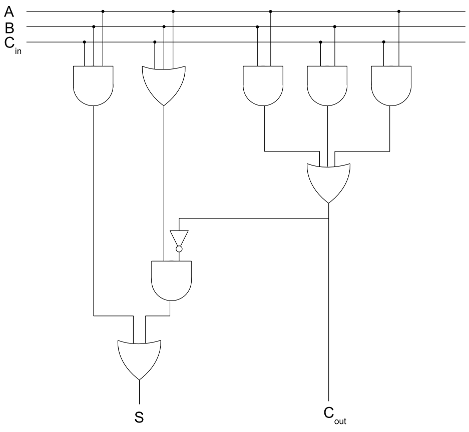
X	Y	$X \vee Y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT is logical negation:

X	$\neg X$
1	0
0	1

Gates and logic circuits

Figure 1: Binary adder



Any Boolean expression can be written as a logical circuit network using *gates*. Table 1 lists the gates and their equivalent Boolean operations.

Consider the following pair of expressions:

$$(A \wedge B \wedge C_{in}) \vee \{(A \vee B \vee C_{in}) \wedge \\ \neg\left[(A \wedge B) \vee (B \wedge C_{in}) \vee (A \wedge C_{in})\right]\}$$

$$(A \wedge B) \vee (B \wedge C_{in}) \vee (A \wedge C_{in})$$

This is a *binary adder* and the equivalent logic circuit is shown in Figure 1. The first expression is the sum (S) and the second is the *carry bit* (C_{out}). Note how the circuit allows repeated expressions to be re-used without needing multiple versions.

A set of adders lined up in series with each “carry out” bit (C_{out}) routed to the next “carry in” bit (C_{in}) can be used to add multiple digit binary numbers.

Tautologies and logical implication

A *tautology* is an expression that is always true. Consider for instance:

$$P \vee \neg P$$

Now, consider these two new operators: logical implication or if-then ($\rightarrow$) and if-and-only-if ($\iff$).

If X then Y ($X \rightarrow Y$) is defined:

X	Y	$X \rightarrow Y$
0	0	1
0	1	1
1	0	0
1	1	1

This is equivalent to $\neg X \vee Y$.

To define if-and-only-if, we first define the exclusive OR or XOR which is as follows:

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

If-and-only-if, $\iff$, is simply a negated XOR ($\neg(X \oplus Y)$).

Now consider two equivalent expressions, for instance $X \rightarrow Y$ and $\neg X \vee Y$, above. We can write:

$$(X \rightarrow Y) \iff (\neg X \vee Y)$$

This expression is a tautology. It also encapsulates a *rule-of-inference*, that is you can substitute the expression on the left for the one on the right and vice versa.

For the one way case, where you can substitute the expresson on the left for the one on the right, but not the other way around, we use if-then. For example:

$$(X \iff Y) \to (X \to Y)$$

Boolean logic and computational complexity

Boolean algebra is important in computational complexity theory. Satisfying a Boolean expression is a prototypical NP-complete problem. The worst-case running time to find a set of inputs that returns a value of 1 increases exponentially with the number of inputs. Moreover, other NP problems are equivalent to this problem and all such equivalent problems are "NP-complete".

Note, however, that NP does not mean "non-polynomial", but rather, "non-deterministic polynomial". In other words, an NP problem will always run in polynomial time on a *non-deterministic computer*. A non-deterministic computer is a theoretical construct such that every time it encounters a branch, the computer takes both simultaneously, so it has the property of always finding the optimal path in any given algorithm.

The most discussed NP-complete problem is the travelling salesmen problem, which is as follows: given a set of points, what is the shortest path that passes through all of them? Note that in this context, it must be formulated as a decision problem: that is, given a set of points and a distance threshold, find a path that is less than the threshold. Presumably, one could design a network that would accurately model any given travelling salesment problem.

While any potential solution can be verified in polynomial time–the running time is proportional to the size of the network–finding a solution can currently only be acheived in NP time–the worst-case running time is proportional the exponent of the size. It is currently an unsolved problem whether algorithms exist that can find a solution to NP-complete problems in strict polynomial time.

Exercises

1. (a) Derive the truth table for NOR and NAND.

   (b) Write all the gates and operators in terms of NOT and NOR.

   (c) Write all the gates and operators in terms of NOT and NAND.

2. (a) Derive the truth table for the binary adder. Hint: here I've started it off for you,

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

   (b) See if you can come up with your own formulation for the adder. In particular, try to minimize the number of gates. The example uses a total of nine (9) gates. Can you invent one that uses fewer?

3. Prove the following tautologies (hint: once you have proven one tautology, you can use it to help prove the others):

   (a) $\neg(A \wedge B) \iff (\neg A \vee \neg B)$

   (b) $\neg(A \vee B) \iff (\neg A \wedge \neg B)$

   (c) $[A \wedge (B \vee C)] \iff [(A \wedge B) \vee (A \wedge C)]$

   (d) $[A \vee (B \wedge C)] \iff [(A \vee B) \wedge (A \vee C)]$

   (e) $[(A \to B) \wedge (B \to A)] \iff (A \iff B)$