An Introduction to Machine Learning

Harrison B. Prosper

Florida State University

YETI 19, Durham University, UK

7 January, 2019

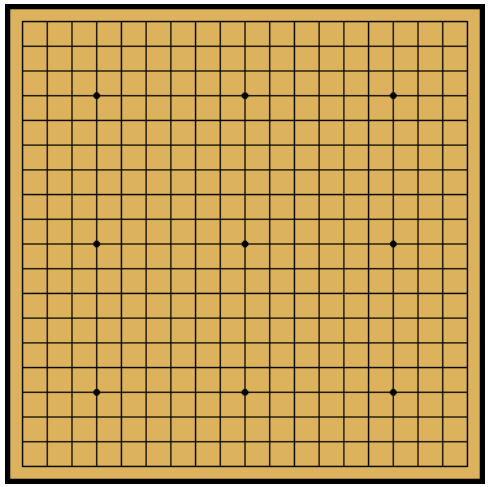
Topics

- Introduction
- A Bit of Theory
- Boosted Decision Trees
- Deep Neural Networks
- The Future of Machine Learning

INTRODUCTION

AlphaGo 4, Homo Sapiens 1

2016 – Google's AlphaGo program beats Go champion Lee Sodol.



Photograph: Yonhap/Reuters

ARTICLE

doi:10.1038/nature24270

Mastering the game of Go without human knowledge

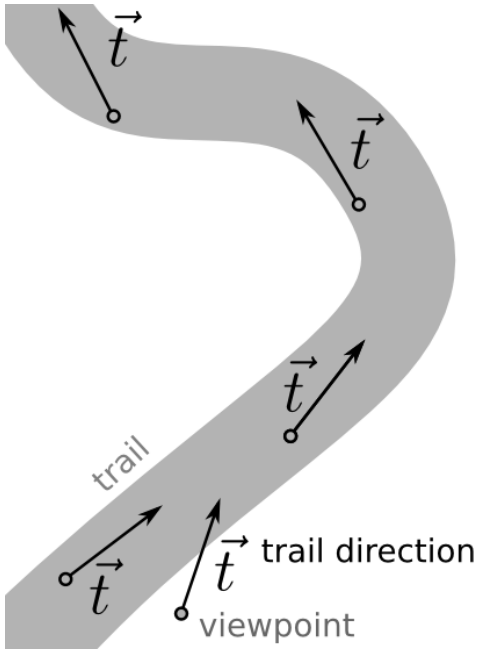
David Silver[1]*, Julian Schrittwieser[1]*, Karen Simonyan[1]*, Ioannis Antonoglou[1], Aja Huang[1], Arthur Guez[1], Thomas Hubert[1], Lucas Baker[1], Matthew Lai[1], Adrian Bolton[1], Yutian Chen[1], Timothy Lillicrap[1], Fan Hui[1], Laurent Sifre[1], George van den Driessche[1], Thore Graepel[1] & Demis Hassabis[1]

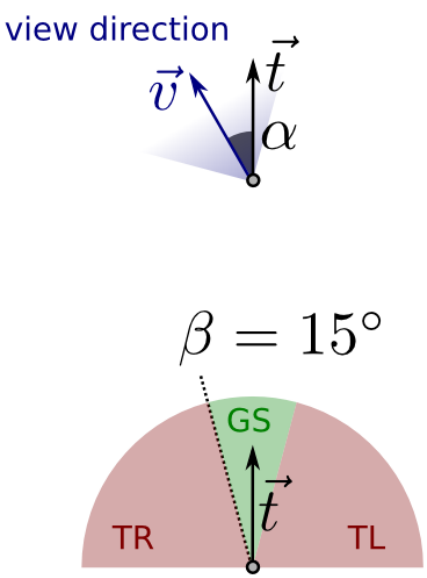
A long-standing goal of artificial intelligence is an algorithm that learns, *tabula rasa*, superhuman proficiency in challenging domains. Recently, AlphaGo became the first program to defeat a world champion in the game of Go. The tree search in AlphaGo evaluated positions and selected moves using deep neural networks. These neural networks were trained by supervised learning from human expert moves, and by reinforcement learning from self-play. Here we introduce an algorithm based solely on reinforcement learning, without human data, guidance or domain knowledge beyond game rules. AlphaGo becomes its own teacher: a neural network is trained to predict AlphaGo's own move selections and also the winner of AlphaGo's games. This neural network improves the strength of the tree search, resulting in higher quality move selection and stronger self-play in the next iteration. Starting *tabula rasa*, our new program AlphaGo Zero achieved superhuman performance, winning 100–0 against the previously published, champion-defeating AlphaGo.

Follow the Yellow Brick Road!

Giusti *et al.* treat the problem of trail navigation as a classification problem!

Data: 8 hours of 1920 x 1080 30fps video using 3 GoPro cameras.





IEEE Robotics and Automation Letters (Volume: 1, Issue: 2, July 2016)

What is Machine Learning?

The use of computer-based algorithms for constructing useful *models* of data.

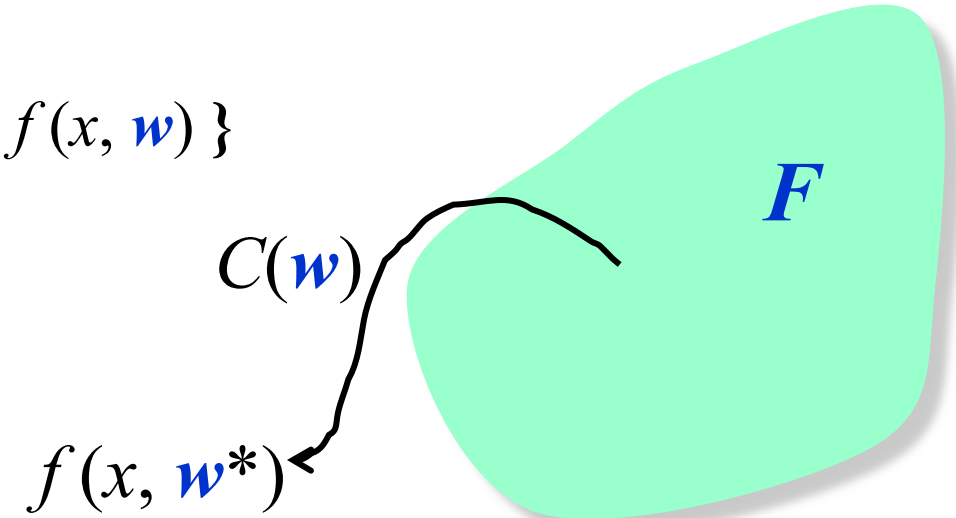
Machine learning algorithms fall into five broad categories:

1. Supervised Learning
2. Semi-supervised Learning
3. Unsupervised Learning
4. Reinforcement Learning
5. Generative Learning

Machine Learning

Choose

Function space	$F = \{ f(x, w) \}$
Constraint	C
Loss function	L



The Goal

Choose a suitable function from the function space. The suitability of the function is assessed with a loss function, (e.g., the quadratic loss $L(y, f) = (y - f)^2$), which quantifies the *cost* of making a *bad* choice.

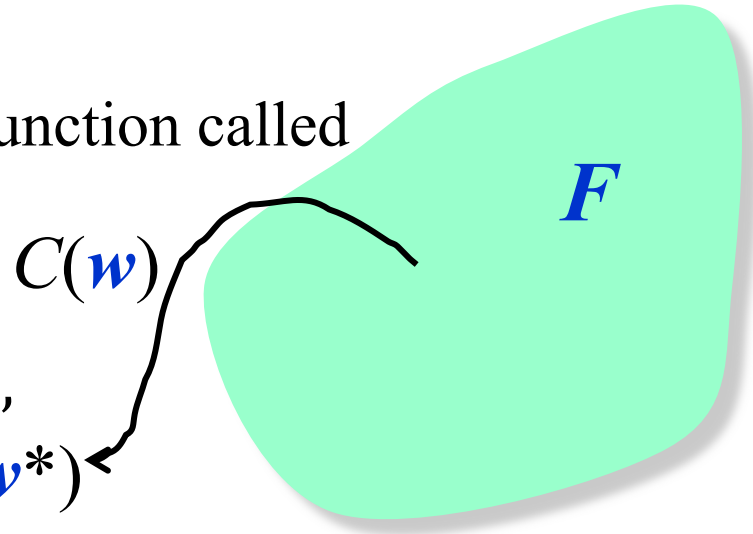
Machine Learning

Method

Find $f(x, \boldsymbol{w}^*)$ by minimizing a function called the (constrained) empirical risk

$$R(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f_i) + C(\boldsymbol{w}),$$

$$f_i \equiv f(x_i, \boldsymbol{w})$$



The constraint $C(\boldsymbol{w})$ guides the choice of $f(x, \boldsymbol{w})$.

Warning: in the machine learning world, $R(\boldsymbol{w})$ is often referred to as the loss function.

Machine Learning

A great deal of the machine learning research is devoted to finding highly flexible functions $f(x, w)$ that render the minimization of

$$R(w) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f_i) + C(w)$$

computationally feasible.

There is also quite a bit of effort devoted to exploring different loss functions and constraints.

A BIT OF THEORY

Minimizing the Average Loss

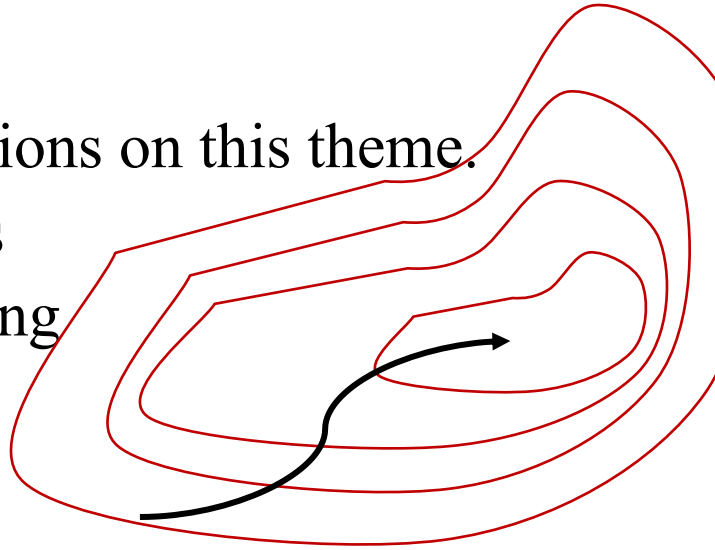
A loss function defines a "landscape" in the space of parameters, or equivalently in the *space of functions*.

The goal is to find the lowest point in that landscape, by moving in the direction of the local *negative* gradient:

$$w_i \leftarrow w_i - \rho \frac{\partial R(w)}{\partial w_i}$$

Most minimization algorithms are variations on this theme.

Stochastic Gradient Descent (SGD) uses random subsets (batches) of the training data to provide *noisy* estimates of the gradient.



Minimizing the Average Loss

Consider the average *quadratic loss* in the limit $N \to \infty$

$$R(w) = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i, w))^2 + C(w)$$

$$\to \int dx \int dy \left(y - f(x, w)\right)^2 p(y, x)$$

$$= \int dx\, p(x)\left[\int dy\, (y - f)^2\, p(y|x)\right]$$

where $p(y|x) = p(y, x)/p(x)$ and where the influence of the constraint (in this limit) is assumed to be negligible.

Note, R is a *functional* of $f(x, w)$, that is, R depends on infinitely many values of $f(x, w)$.

Minimizing the Average Loss

If we change the function $\boldsymbol{f}$ by a small *arbitrary* function $\boldsymbol{\delta f}$ a small change

$$\boldsymbol{\delta R} = 2 \int dx\, p(x) \boldsymbol{\delta f} \left[\int dy (y - f) p(y|x) \right]$$

will be induced in R. In general, $\boldsymbol{\delta R} \neq 0$.

But, if the function $\boldsymbol{f}$ is flexible enough then we shall be able to reach the minimum of R, where $\boldsymbol{\delta R} = 0$.

But, the only way to guarantee that $\boldsymbol{\delta R} = 0$ for all $\boldsymbol{\delta f}$ and for all $\boldsymbol{x}$ is if the quantity in brackets is *zero*, that is, if

$$f(x) = \int y\, p(y \mid x)\, dy$$

Classification

Recall that Bayes' theorem is

$$p(y|x) = \frac{p(y,x)}{p(x)} = \frac{p(x\,|y)\,p(y)}{\int p(x\,|y)\,p(y)dy}$$

Now assign the target value $y = 1$ to objects of class ***s*** and target value $y = 0$ to objects of class ***b***.

Then

$$f(x) = \int y\,p(y\mid x)\,dx = p(1|x)$$

$$\equiv p(\boldsymbol{s}|x)$$

That is, the function $f(x)$ equals the class probability.

Classification

In 1990*, the result

$$f(x) = p(s|x) = \frac{p(x,s)}{p(x)} = \frac{p(x|s)p(s)}{p(x|s)p(s) + p(x|b)p(b)}$$

was derived in the context of neural networks.

But notice our discussion so far made no mention of neural networks!

* Ruck et al., *IEEE Trans. Neural Networks* 4, 296-298 (1990); Wan, *IEEE Trans. Neural Networks* 4, 303-305 (1990);
Richard and Lippmann, *Neural Computation.* 3, 461-483 (1991)

Classification

The point is that the result

$$f(x) = p(s|x) = \frac{p(x,s)}{p(x)}$$

depends only on the *form* of the loss function *provided that*:

1. the training data are sufficiently numerous,
2. the function $f(x, \boldsymbol{w})$ is sufficiently flexible, and
3. the minimum of R can be found.

In particular,

the result does not depend on the nature of the function $f(x, w)$.

Classification

Note, if $p(s) = p(b)$, we arrive at the discriminant

$$D(x) = \frac{p(x|s)}{p(x|s) + p(x|b)} \equiv \frac{s(x)}{s(x) + b(x)}$$

which is an extremely useful result because it suggests many potential machine learning applications.

Example: given that $s(x) = \frac{D}{1-D} b(x)$, the density $s(x)$ can be approximated as a correction to a known function $b(x)$.

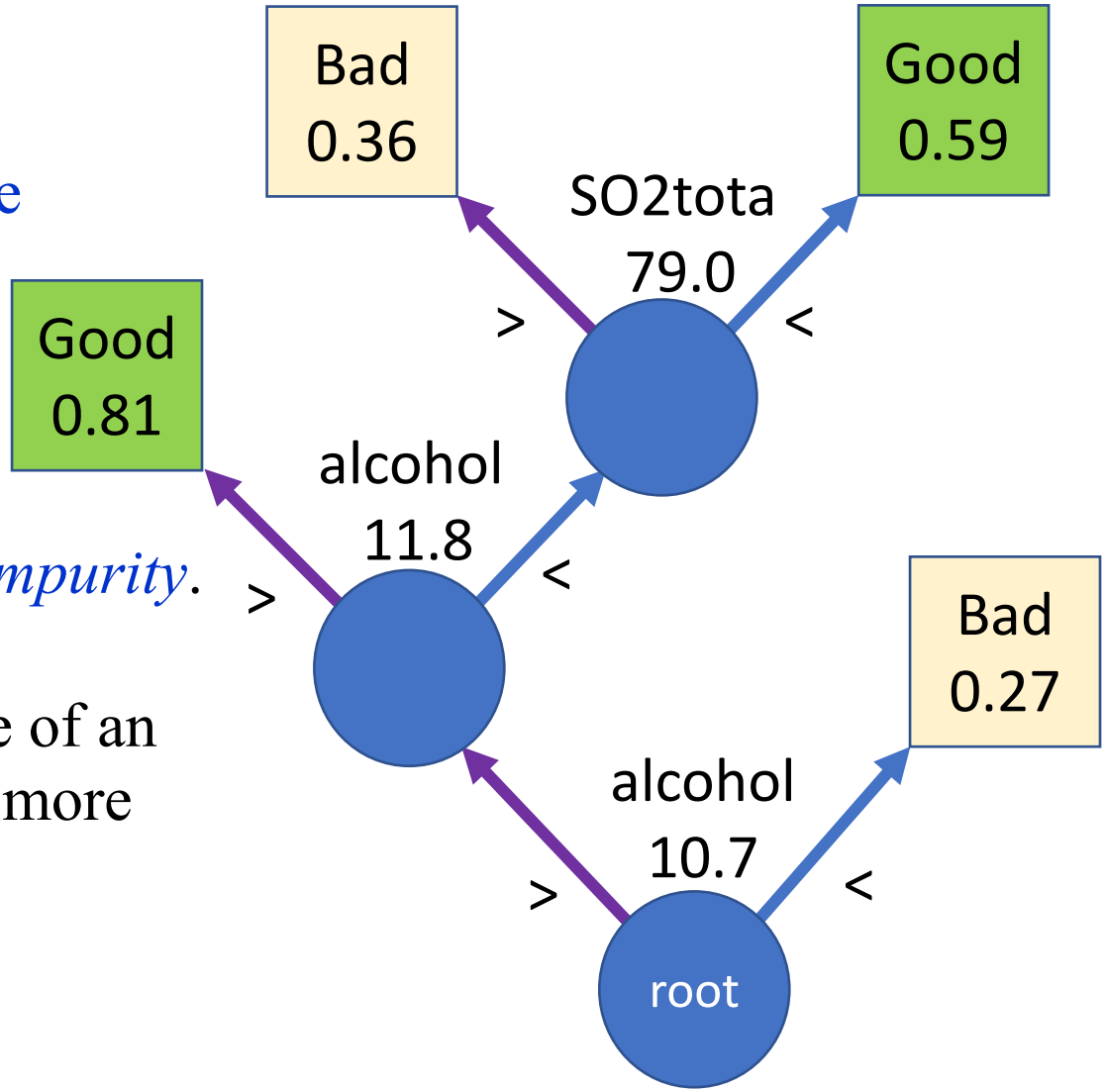
BOOSTED DECISION TREES

Decision Trees

Decision tree:
a sequence of if then else statements.

Basic idea: recursively partition the space into regions of diminishing *impurity*.

This is a simple example of an automated wine taster…more later…!

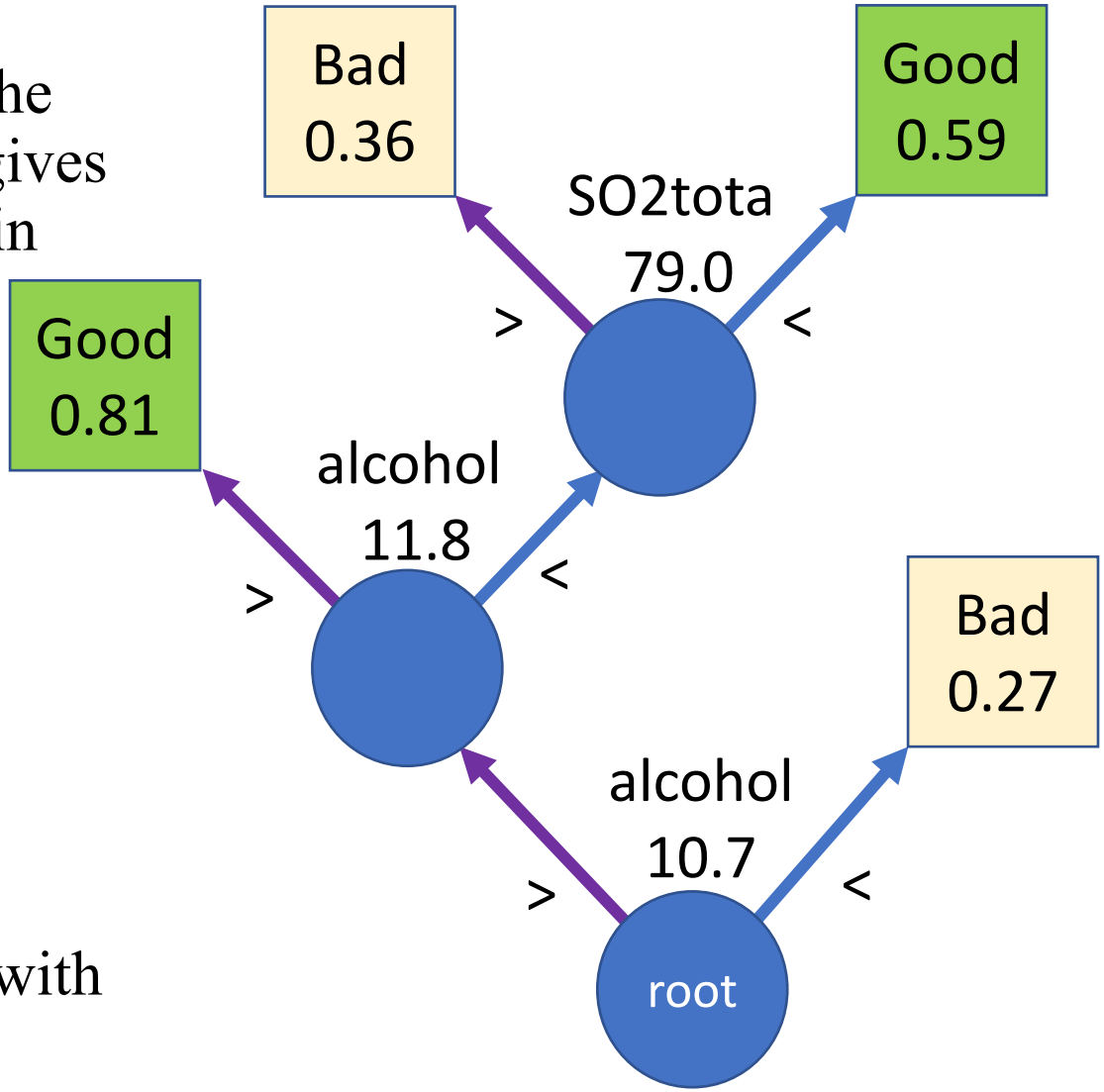


Decision Trees

For each variable, find the partition ("cut") that gives the greatest *decrease* in impurity:

- **Impurity** (parent)
  - **Impurity** ("left")
  - **Impurity** ("right")

Then, choose the best partition among all partitions, and repeat with each child.



Decision Trees

The most common impurity measure is the Gini index (Corrado Gini, 1884-1965):

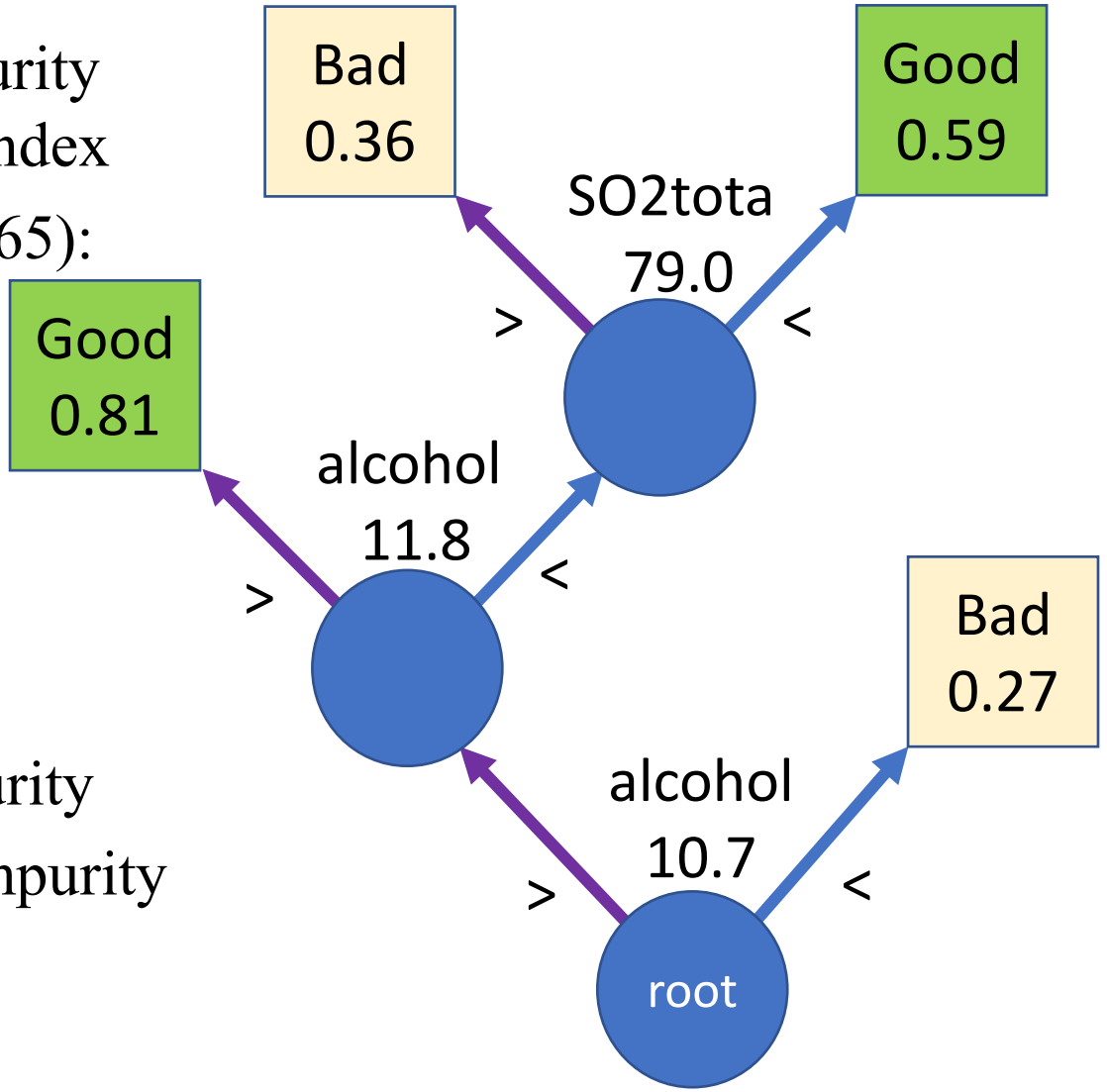
Gini index = $p\,(1 - p)$

where p is the purity

$p = S / (S + B)$

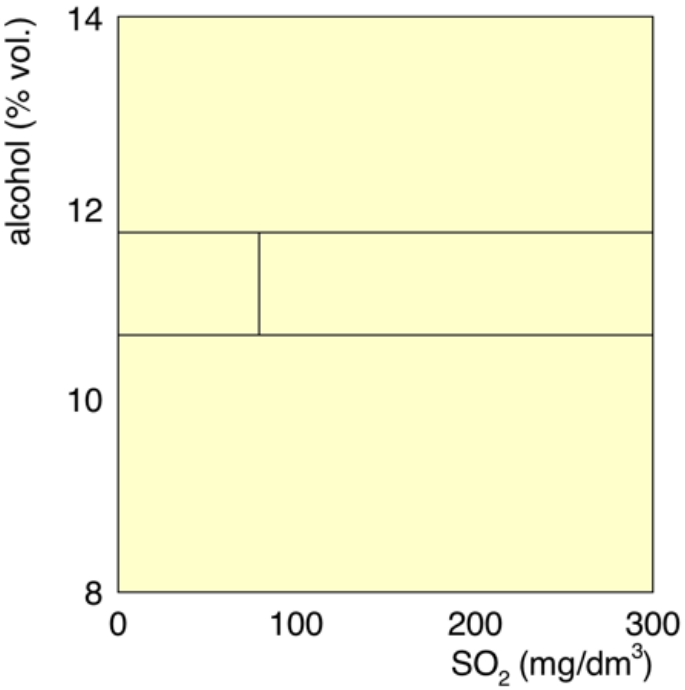
$p = 0$ or 1 = maximal purity

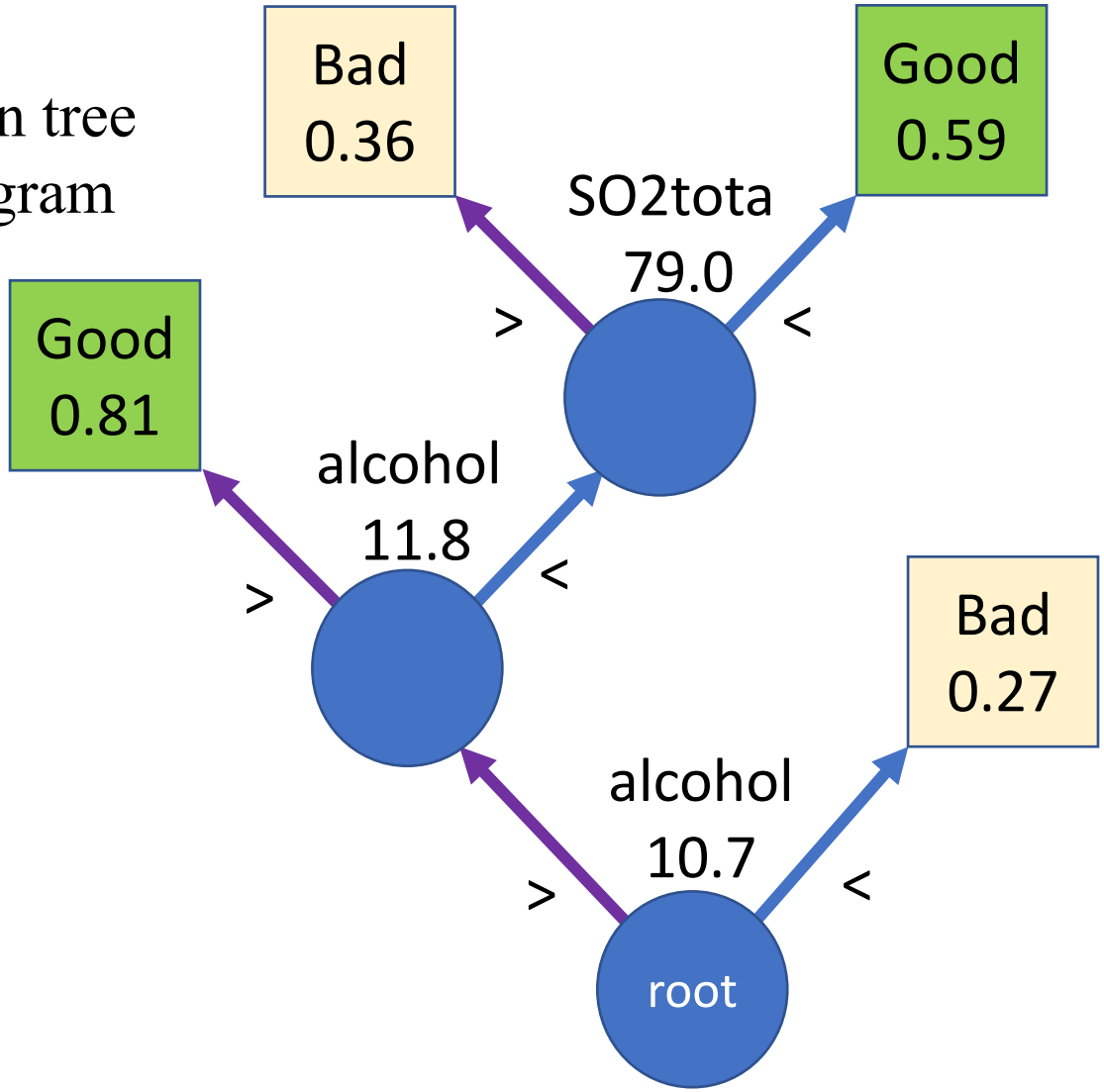
$p = 0.5$ = maximal impurity



Decision Trees

Geometrically, a decision tree is a *d-dimensional* histogram in which the bins are created *recursively*.





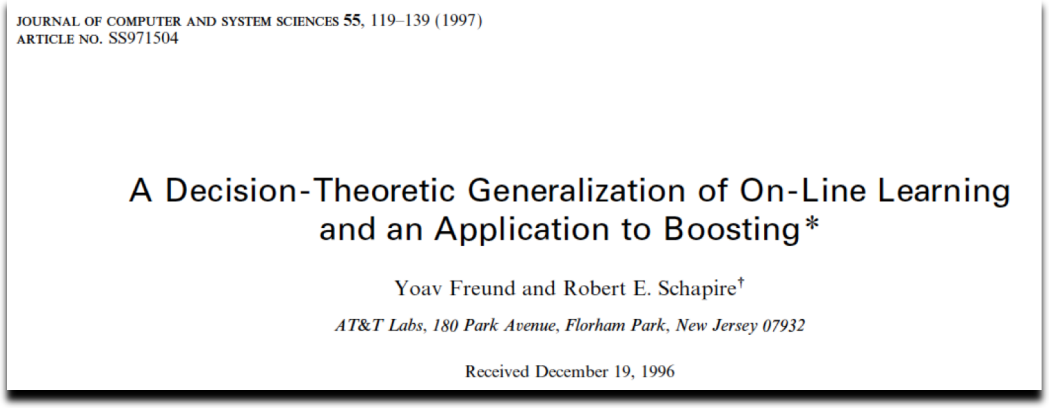
Decision Trees

Unfortunately, decision trees are unstable!

A Silk Purse from a Sow's Ear!

In 1997, AT&T researchers Freund and Schapire [Journal of Computer and Sys. Sci. **55** (1), 119 (1997)] showed that it was possible to build highly effective classifiers by combining a large number of mediocre ones!

The Freund-Schapire algorithm, which they called AdaBoost, was the first successful method to *boost* (i.e., enhance) the performance of poorly performing classifiers by *averaging* their outputs.



JOURNAL OF COMPUTER AND SYSTEM SCIENCES 55, 119–139 (1997)
ARTICLE NO. SS971504

A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*

Yoav Freund and Robert E. Schapire†

AT&T Labs, 180 Park Avenue, Florham Park, New Jersey 07932

Received December 19, 1996

A Silk Purse from a Sow's Ear!

The most popular methods (used mostly with decision trees) are:

- Bagging: each tree is trained on a bootstrap* sample drawn from the training set
- Random Forest: bagging with randomized trees
- Boosting: each tree trained on a different reweighting of the training set

*A bootstrap sample is a sample of size N drawn, *with replacement*, from another of the same size. Duplicates can occur and are allowed.

EXAMPLE: WINE TASTING!

Wine Tasting

Wine tasting is big business. But, can a machine do it?

In principle, yes, if we can establish the physical attributes that define "good" wine, such as this one for **$117,000** a bottle!

Wine Tasting

We'll use AdaBoost to build a classifier to distinguish good wines from bad wines

from Vinho Verde in Portugal.

Wine Tasting

Let's define a good wine as one with expert rating ≥ 0.6 on a scale from 0 to 1, where **1** is a wine from Heaven and **0** is a wine from Hell!

We'll use data from

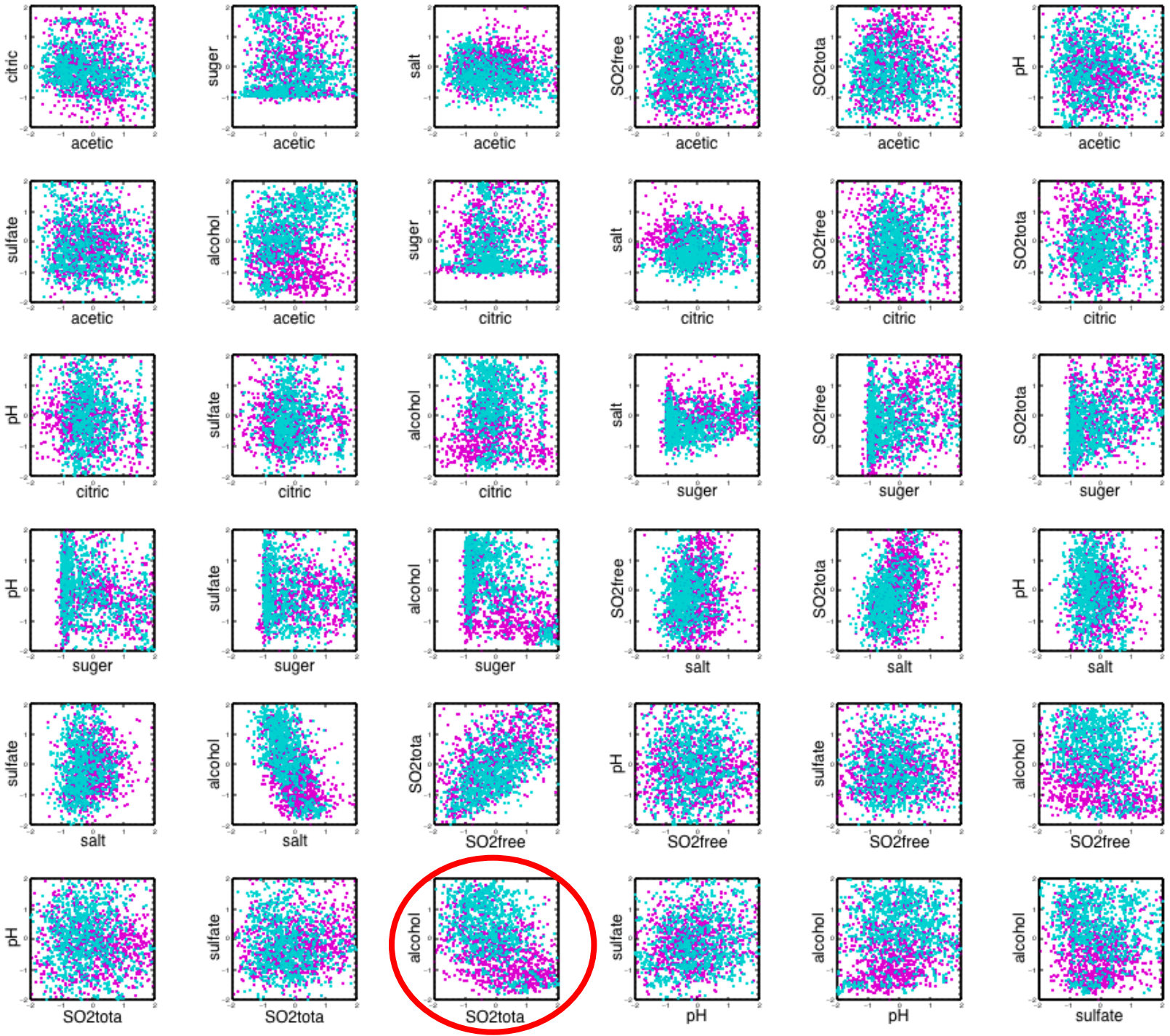
Cortez *et al.**

* P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.
Modeling wine preferences by data mining from physicochemical properties.
In Decision Support Systems, Elsevier, 47(4):547-553. ISSN: 0167-9236

Wine Tasting: Data

Data: [Cortez *et al.*, 2009].

variables	description
acetic	acetic acid
citric	citric acid
sugar	residual sugar
salt	NaCl
SO2free	free sulfur dioxide
SO2tota	total sulfur dioxide
pH	pH
sulfate	potassium sulfate
alcohol	alcohol content
quality	(between 0 and 1)

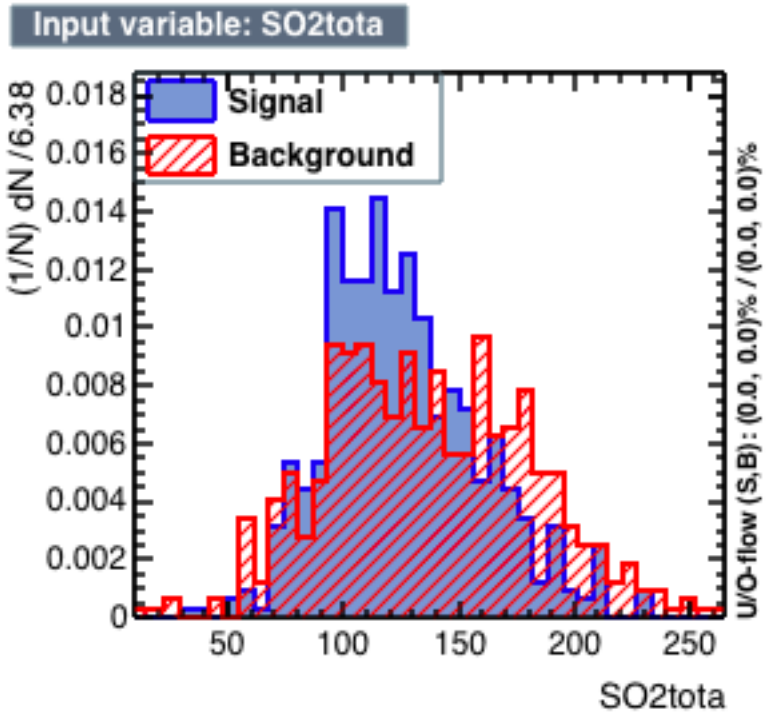

citric acetic
suger acetic
salt acetic
SO2free acetic
SO2tota acetic
pH acetic
sulfate acetic
alcohol acetic
suger citric
salt citric
SO2free citric
SO2tota citric
pH citric
sulfate citric
alcohol citric
salt suger
SO2free suger
SO2tota suger
pH suger
sulfate suger
alcohol suger
SO2free salt
SO2tota salt
pH salt
sulfate salt
alcohol salt
SO2tota SO2free
pH SO2free
sulfate SO2free
alcohol SO2free
pH SO2tota
sulfate SO2tota
alcohol SO2tota
sulfate pH
alcohol pH
alcohol sulfate

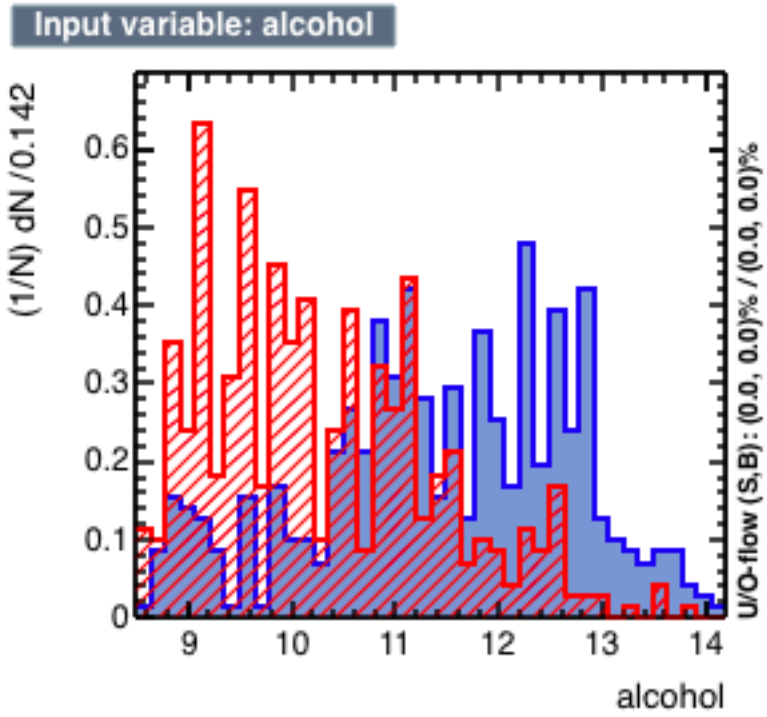
Wine Tasting: Variables

Variables:

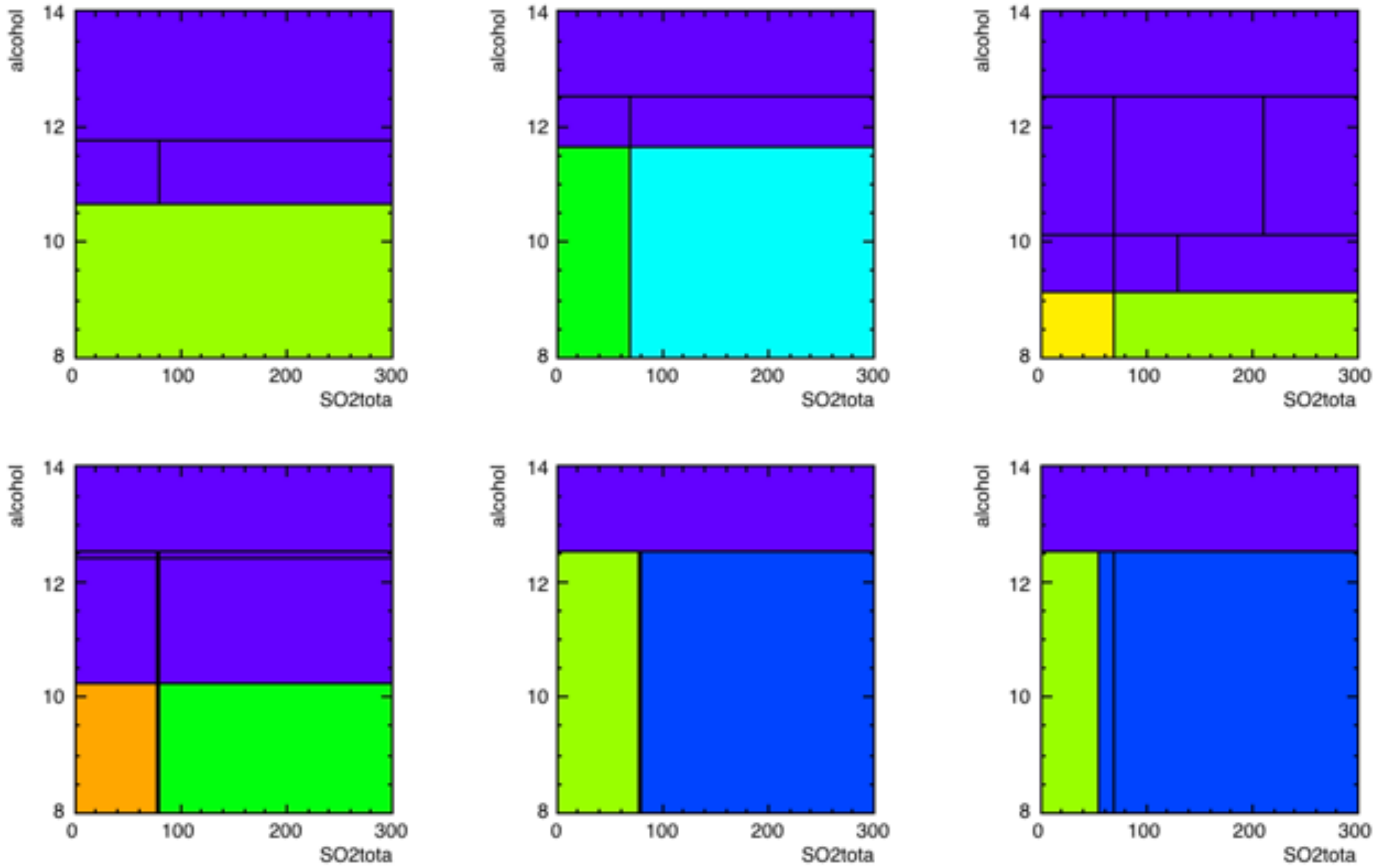
SO2tota: the total sulfur dioxide content (mg/dm^3)

alcohol: alcohol content (% volume)





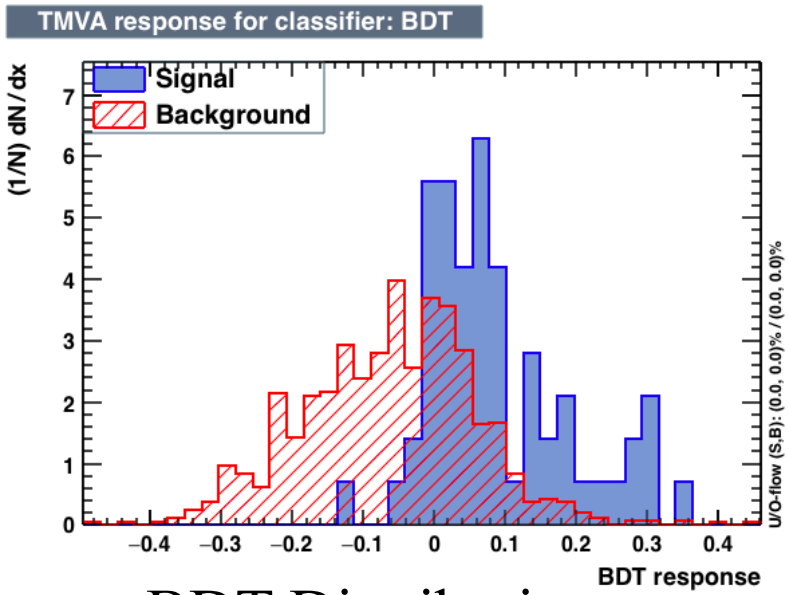
Wine Tasting: First 6 Decision Trees



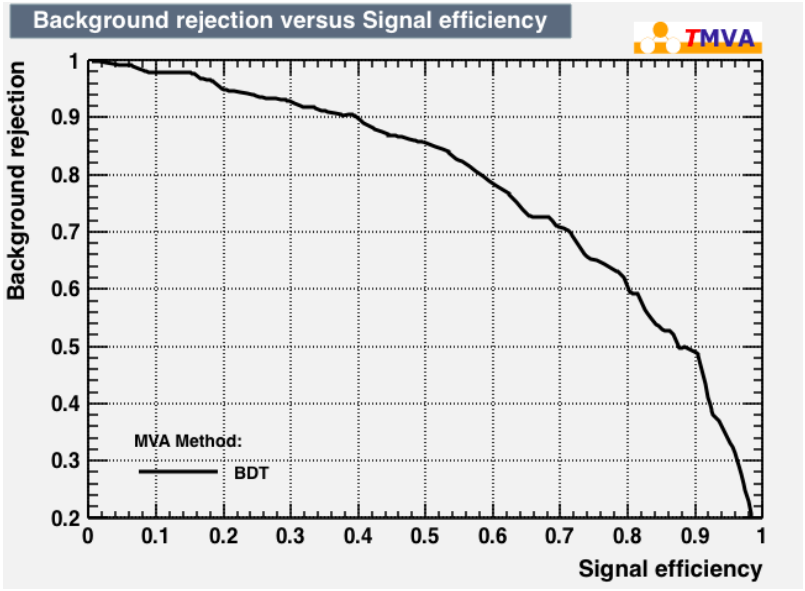
Wine Tasting: Results

x = SO2tota

y = alcohol



BDT Distribution

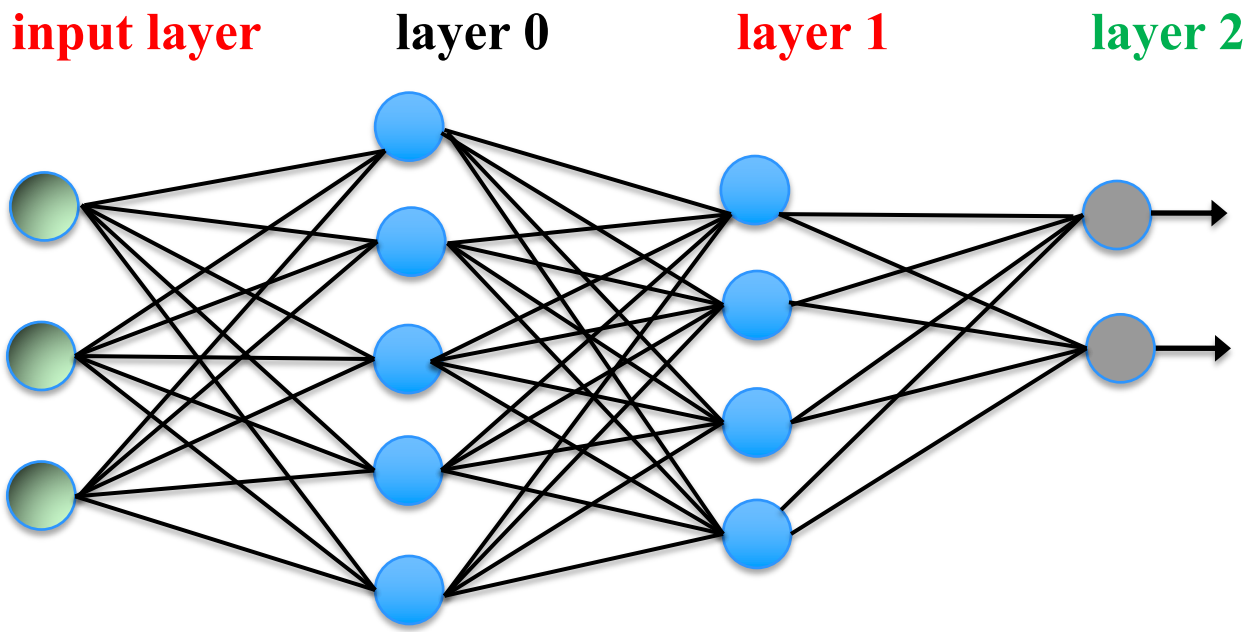


Fraction of bad wine rejected for a given fraction of good wine accepted.

$$BDT(x,y) = \sum_{k=0}^{99} \alpha_k f(x,y,w_k)$$

DEEP NEURAL NETWORKS

Deep Neural Networks



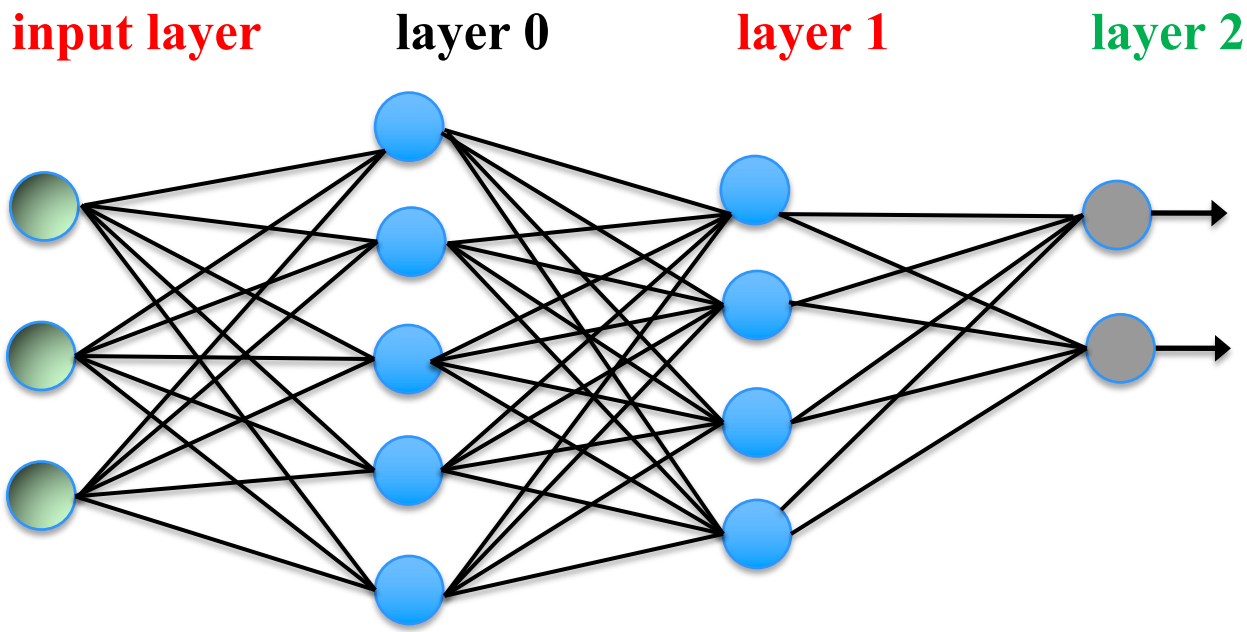
$\boldsymbol{x}$

$$\boldsymbol{h}_1 = \boldsymbol{h}(\mathbf{b}_0 + \mathbf{w}_0\boldsymbol{x})$$

$$\boldsymbol{h}_2 = \boldsymbol{h}(\mathbf{b}_1 + \mathbf{w}_1\boldsymbol{h}_1)$$

$$\mathbf{o} = \boldsymbol{g}(\mathbf{b}_2 + \mathbf{w}_2\boldsymbol{h}_2)$$

Deep Neural Networks



A 3-layer DNN

$$o = g(\boldsymbol{b_2} + \boldsymbol{w_2} h(\boldsymbol{b_1} + \boldsymbol{w_1} h(\boldsymbol{b_0} + \boldsymbol{w_0} x)))$$

$h(z)$ = ReLU(z) [= max(0, z)], tanh(z)

$g(z)$ = Identity(z), logistic(z) = 1/[1 + exp(-z)]

Deep Neural Networks

- In 2006, University of Toronto researchers Hinton, Osindero, and Teh (HOT*) succeeded in training a deep neural network for the first time. Each layer was trained to produce a representation of its inputs that served as the training data for the next layer. Then the entire network was adjusted using gradient descent.

- This breakthrough seemed to provide compelling evidence that the training of deep neural networks requires careful initialization of parameters and sophisticated machine learning algorithms.

* Hinton, G. E., Osindero, S. and Teh, Y. (HOT), A fast learning algorithm for deep belief nets, Neural Computation 18, 1527-1554.

Deep Neural Networks

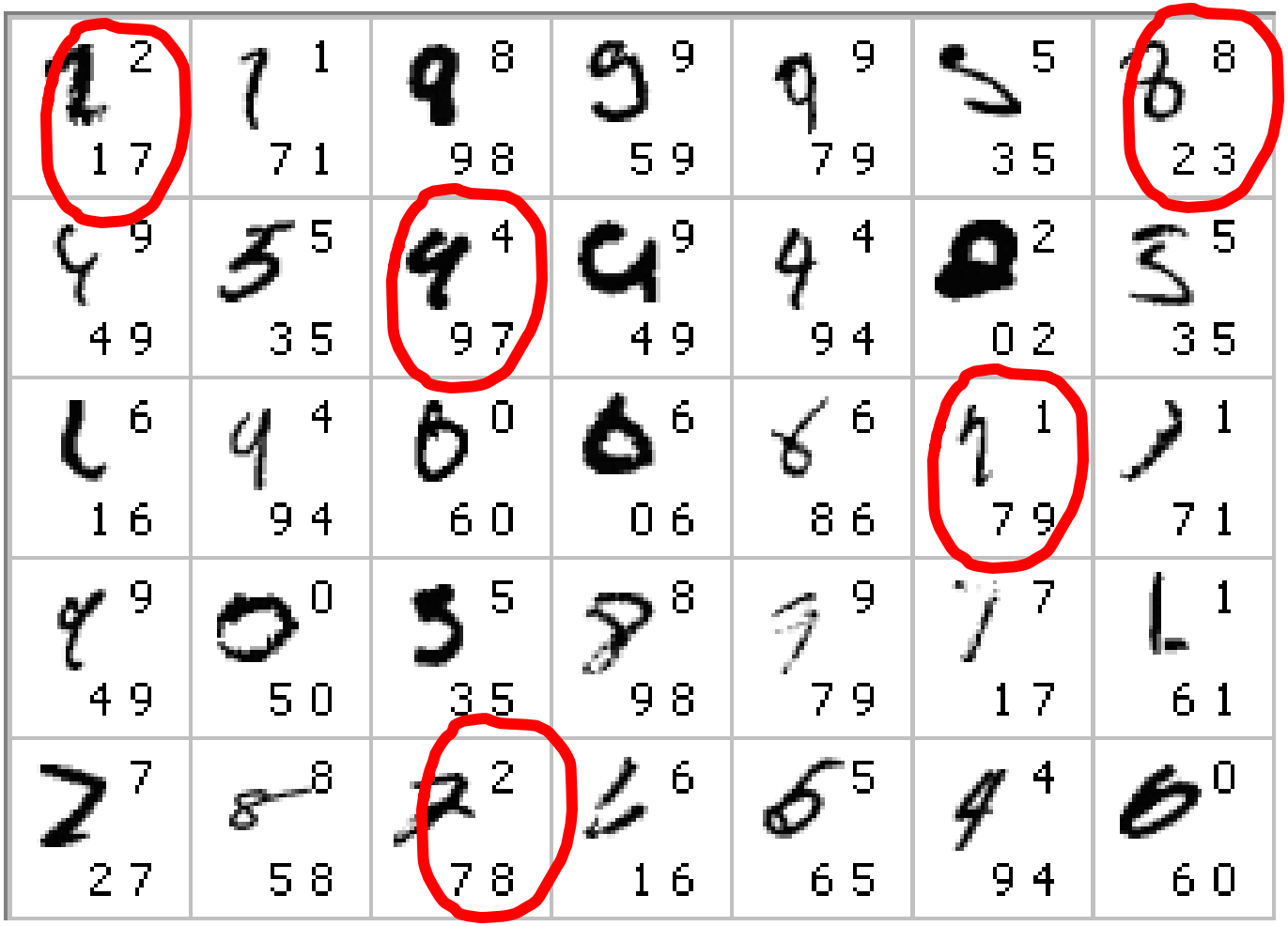
- But, in 2010, Cireşan *et al*.* showed that such cleverness was not needed! The authors succeeded in training a DNN with architecture (**784**, 2500, 2000, 1500, 1000, 500, **10**) that classified the hand-written digits in the MNIST database.

- The database comprises 60,000 $28 \times 28 = 784$ pixel images for training and validation, and 10,000 for testing.

- The error rate of their ~12-million parameter DNN was 35 images out of 10,000. The misclassified images are shown on the next slide.

* Cireşan DC, Meier U, Gambardella LM, Schmidhuber J. , Deep, big, simple neural nets for handwritten digit recognition. Neural Comput. 2010 Dec; 22 (12): 3207-20. http://yann.lecun.com/exdb/mnist/

(784, 2500, 2000, 1500, 1000, 500, 10)

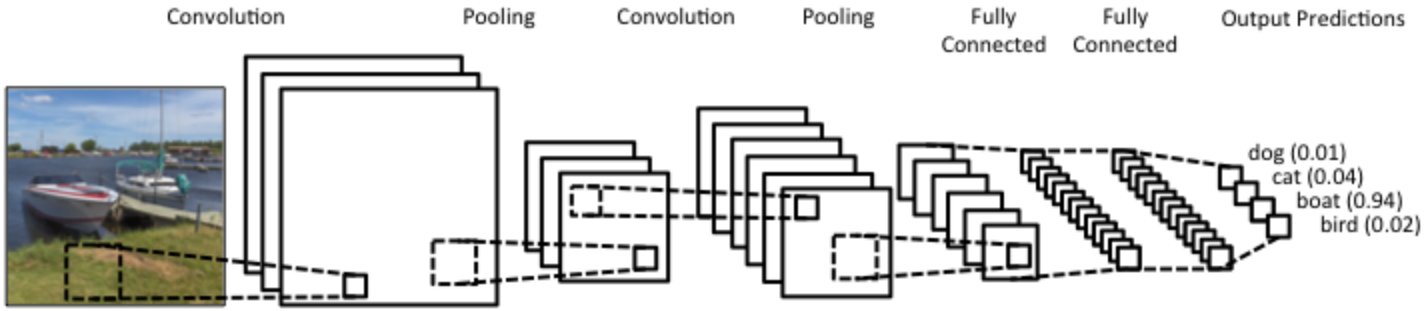


Upper right: correct answer; lower left answer of highest DNN output; lower right answer of next highest DNN output.

Convolutional Neural Networks

Many of the remarkable breakthroughs in tasks such as face recognition use a type of DNN called a convolutional neural network (CNN).

CNNs are *functions* that compress data and classify objects using their compressed representations via a standard fully connected NN. The compression dramatically reduces the dimensionality of the space to be searched.



Source: https://www.clarifai.com/technology

THE FUTURE OF MACHINE LEARNING

Machine Learning and AI

- The most far-reaching application of machine learning is of course artificial intelligence (AI).

- There is a huge amount of hype associated with AI. And we are far from being able to create systems that possess the innate heuristic algorithms that make us human.

- However, it important that we not become complacent because, for better of worse, we do seem to be at the dawn of a revolution based on machine learning-based AI.

McKinsey&Company

MCKINSEY GLOBAL INSTITUTE

A FUTURE THAT WORKS: AUTOMATION, EMPLOYMENT, AND PRODUCTIVITY

JANUARY 2017

EXECUTIVE SUMMARY

"Almost half the activities people are paid almost $16 trillion in wages to do in the global economy have the potential to be automated by adapting currently demonstrated technology, according to our analysis of more than 2,000 work activities across 800 occupations."

McKinsey & Company,

A FUTURE THAT WORKS: AUTOMATION, EMPLOYMENT, AND PRODUCTIVITY

Executive Summary January 2017

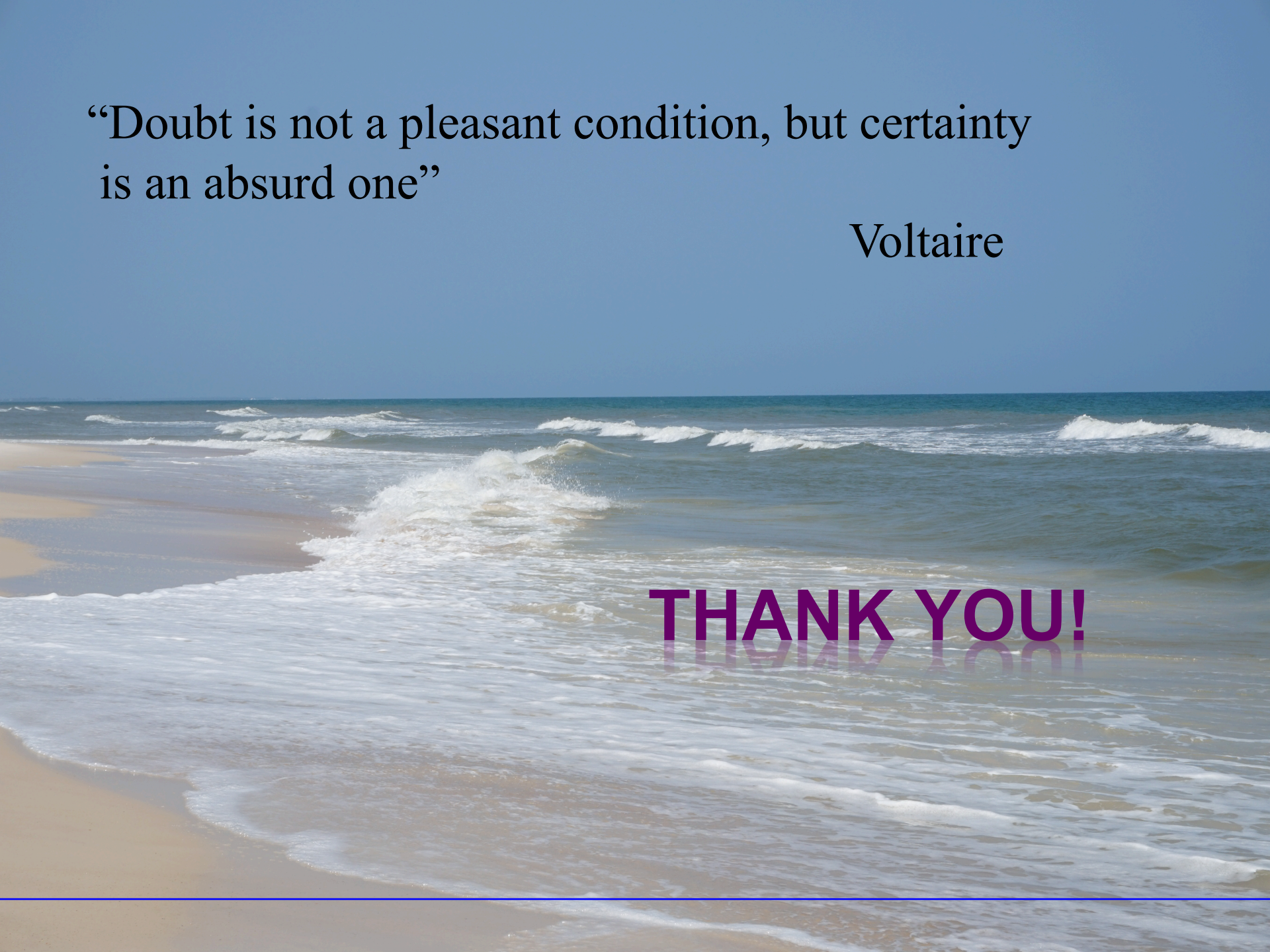
The Future of Machine Learning

By 2050, the following might be in routine use:

1. autonomous personal predictive medical systems
2. autonomous personal tutors
3. autonomous physician's assistant
4. autonomous house servant
5. autonomous pet sitter
6. autonomous vehicles that can drive safely on the M1!

The potential of machine learning and AI is vast and exciting.

But it has been argued (e.g, Bill Gates, Elon Musk, the late Stephen Hawking) that the *dangers* are also vast and fearsome: autonomous, self-aware, drone soldiers, AI drone swarms, AI-enabled computer viruses…


"Doubt is not a pleasant condition, but certainty is an absurd one"
Voltaire
THANK YOU!

Hands-On Exercises

Dependencies

python 2.7.x, x > 9

numpy	array manipulation
pandas	DataFrame manipulation
matplotlib	plotting
scikit-learn	simple machine learning toolkit

Also useful:

scipy	mathematical stuff for scientists
sympy	amazing symbolic algebra package

Installation

git clone https://github.com/hbprosper/YETI

Hands-On Exercises

Exercises

01	wine classification (BDT)
02	wine quality regression (fully connected DNN)
03	Higgs boson VBF vs ggF (fully connected DNN)
04	MNIST hand written digit classification (DNN)

BACKUP

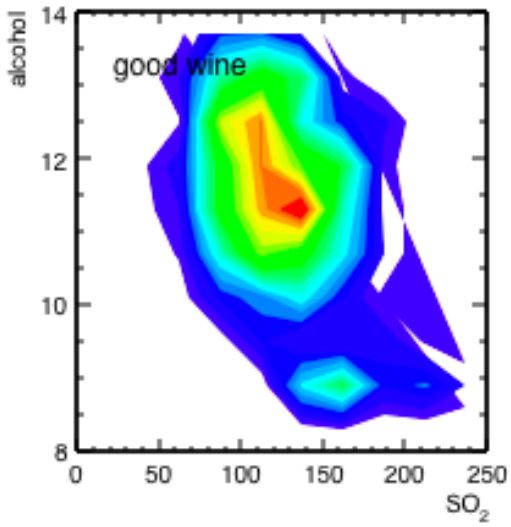
Wine Tasting: Results

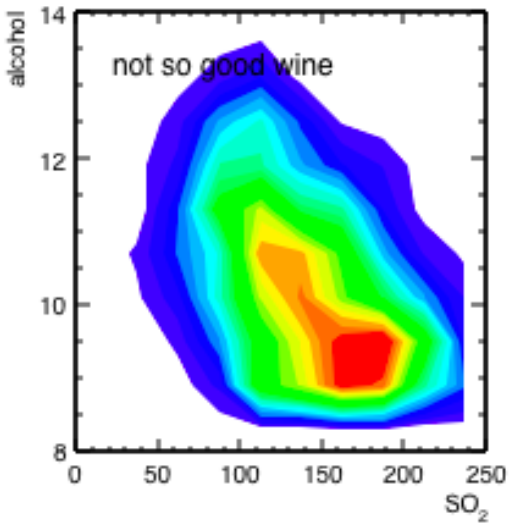
The upper figures are density plots of the training data.

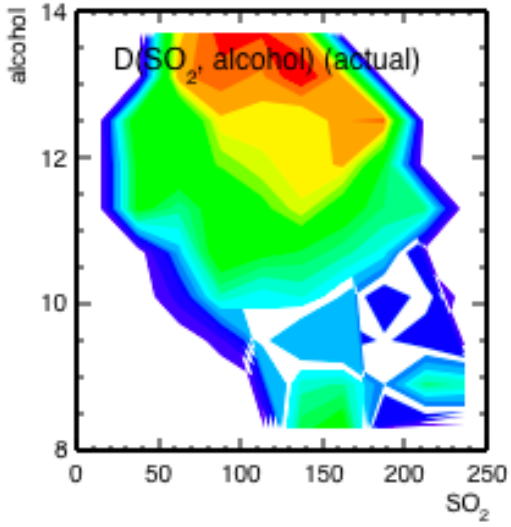
The lower plots are approximations of the discriminant

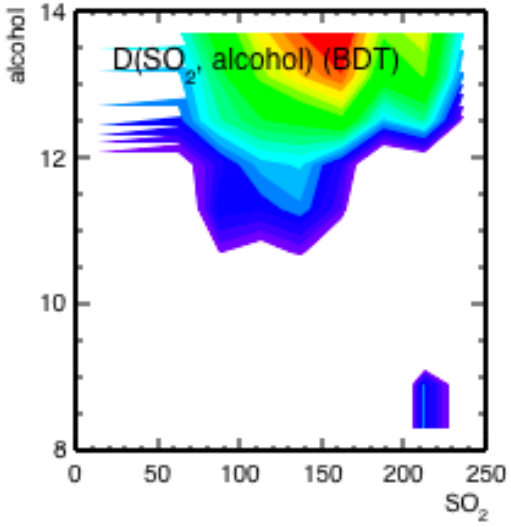
$D(x, y)$

The left, uses 2-D histograms, the right uses the BDT.









Ensemble Methods

Suppose you have an ensemble of classifiers $f(x, w_k)$, which, individually, perform only marginally better than random guessing. Such classifiers are called weak learners.

It is possible to build highly effective classifiers by *averaging* their outputs:

$$f(x) = a_0 + \sum_{n=1}^{N} a_n f(x_n, w_n)$$

Jerome Friedman & Bogdan Popescu (2008)

DEEP NEURAL NETWORKS

Adaptive Boosting

The **AdaBoost** algorithm of Freund and Schapire uses decision trees $f(x, \boldsymbol{w})$ with weights $\boldsymbol{w}$ assigned to each object to be classified, and each assigned a target value of either $y = +1$, or -1, e.g., +1 for signal, –1 for background.

The value assigned to each leaf of $f(x, \boldsymbol{w})$ is also ± 1.

Consequently, for object n, associated with values (y_n, x_n)

$f(x_n, \boldsymbol{w})\, y_n > 0$ for a correct classification

$f(x_n, \boldsymbol{w})\, y_n < 0$ for an incorrect classification

Y. Freund and R.E. Schapire. Journal of Computer and Sys. Sci. **55** (1), 119 (1997)

Adaptive Boosting

Initialize weights $\boldsymbol{w}$ in training set (e.g., setting each to $\mathbf{1/}\boldsymbol{N}$)

for $k = 1$ to K:

1. Create a decision tree $f(x, \boldsymbol{w})$ using the current weights.
2. Compute its error rate $\boldsymbol{\varepsilon}$ on the ***weighted*** training set.
3. Compute $\boldsymbol{\alpha} = \ln(1-\boldsymbol{\varepsilon}) / \boldsymbol{\varepsilon}$ and store as $\boldsymbol{\alpha}_k = \boldsymbol{\alpha}$
4. Update each weight $\boldsymbol{w_n}$ in the training set as follows: $\boldsymbol{w_n} = \boldsymbol{w_n} \exp[-\boldsymbol{\alpha}_k f(x_n, \boldsymbol{w}) y_n] / A$, where A is a normalization constant such that $\sum \boldsymbol{w_n} = 1$. Since $f(x_n, \boldsymbol{w}) y_n < 0$ for an incorrect classification, the weight of misclassified objects is ***increased***.

At the end, compute the average $f(x) = \sum \alpha_k f(x, \boldsymbol{w_k})$

Y. Freund and R.E. Schapire. Journal of Computer and Sys. Sci. **55** (1), 119 (1997)

Adaptive Boosting

AdaBoost is a highly non-intuitive algorithm. However, soon after its invention, Friedman, Hastie and Tibshirani showed that the algorithm is mathematically equivalent to minimizing the following average loss function

$$R(F) = \int \exp(-y\, F(x))\, \boldsymbol{p(x, y)}\, dx\, dy$$

where $F(x) = \sum_{n=1}^{N} a_n\, f(x_n, w_n)$,

Minimizing this loss function yields

$$D(x) = \text{logistic}(2F) = 1/(1 + \exp(-2\, F(x))$$

which can be interpreted as a probability, even though F cannot!

J. Friedman, T. Hastie and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," The Annals of Statistics, 28(2), 377-386, (2000)

A Bit of History: Hilbert's 13th Problem

(One version of Problem 13): Prove

that it is *impossible* to do the following:

$$f(x_1,\ldots,x_n) = F(\, g_1(x_{(1)},\ldots, x_{(m)}),\ldots, g_k(x_{(1)},\ldots, x_{(m)}))$$

for $m < n$ for all n.

In 1957, Kolmogorov proved that it <u>was</u> possible with $m = 3$.

Today, we know that functions of the form

$$f_k(x,w) = \boldsymbol{a_k} + \sum_{j=1}^{H} \boldsymbol{w_{kj}} \boldsymbol{h}\left(\boldsymbol{b_j} + \sum_{i=1}^{I} \boldsymbol{w_{ji}} x_i \right)$$

can provide arbitrarily accurate approximations of real functions of *I* real variables.

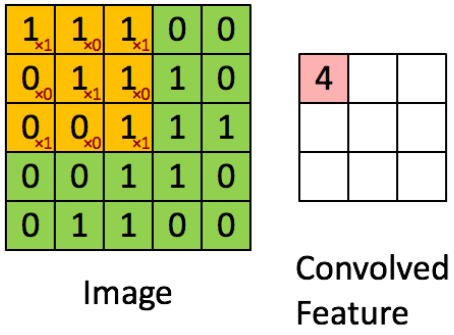
(Hornik, Stinchcombe, and White, *Neural Networks* **2**, 359-366 (1989))

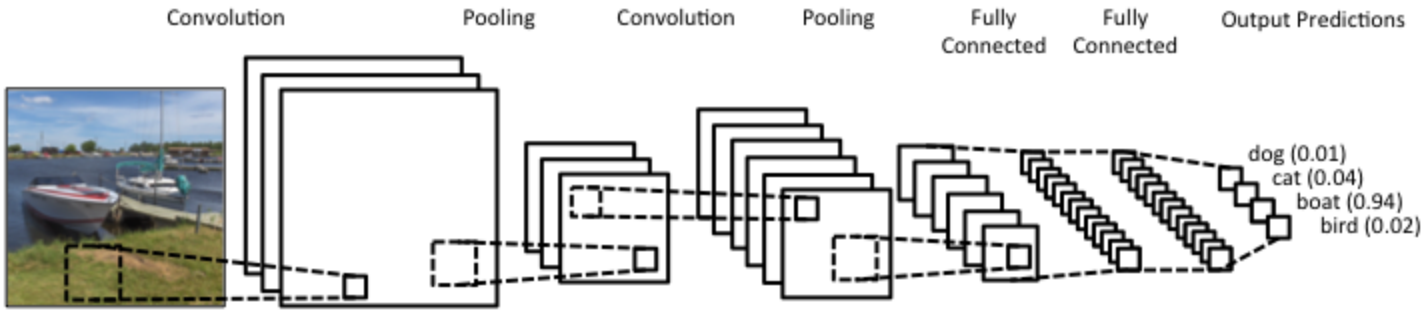
Convolutional Neural Networks

A CNN comprises three types of processing layers: 1. convolution, 2. pooling, and 3. classification.

1. **Convolution layers**

   The input layer is "convolved" with one or more matrices using element-wise products that are then summed. In this example, since the sliding matrix fits 9 times, we compress the input from a 5 x 5 to a to a 3 x 3 matrix.

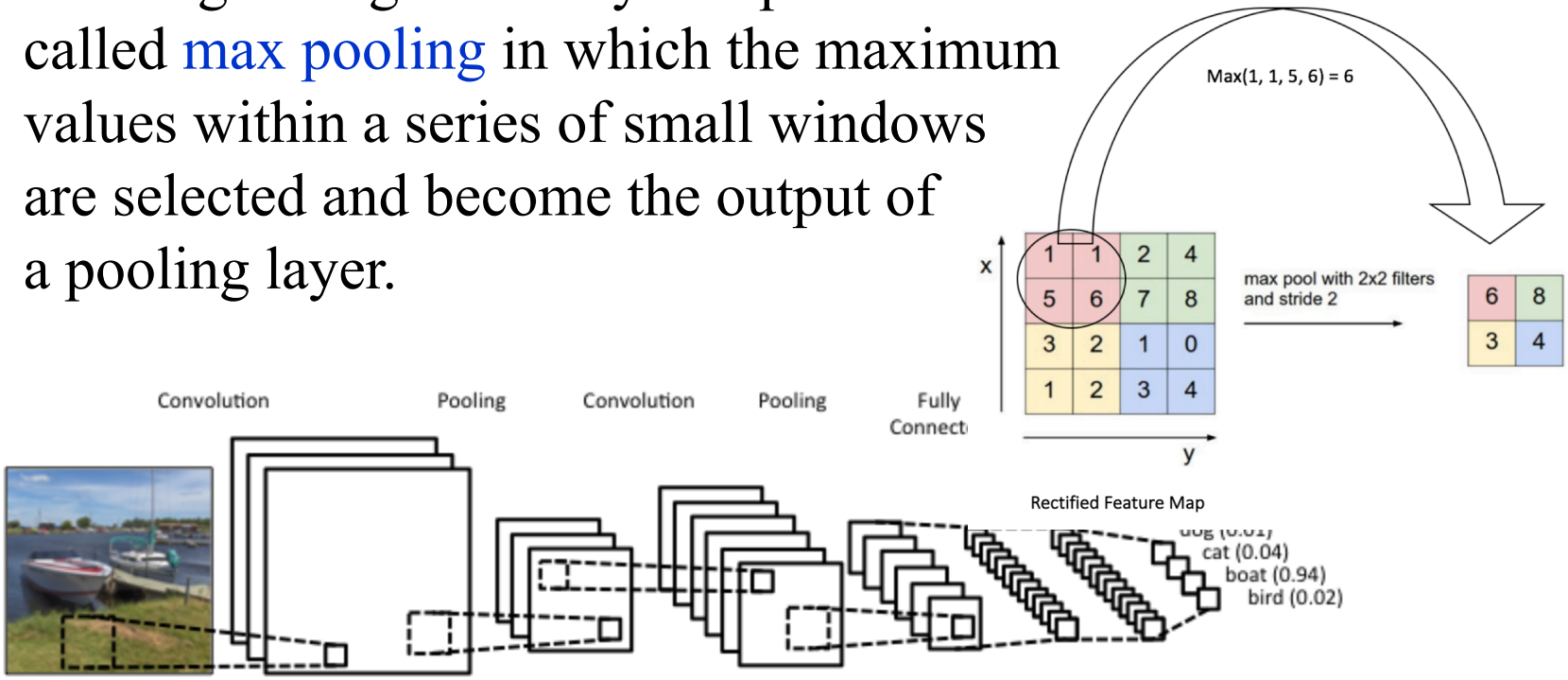




Convolutional Neural Networks

2. Pooling Layers

After convolution, and a pixel by pixel non-linear map (using, e.g., the function $y = \max(0, x) = \text{ReLU}(x)$), a coarse-graining of the layer is performed called max pooling in which the maximum values within a series of small windows are selected and become the output of a pooling layer.



Convolutional Neural Networks

3. Classification Layers

After an alternating sequence of convolution and pooling layers, the outputs go to a standard neural network, either shallow or deep. The final outputs correspond to the different classes and like all flexible classifiers, a CNN approximates,

$$p(C_k|x) = p(x|C_k)p(C_k) / \sum_{m=1}^{M} p(x|C_m)p(C_m)$$