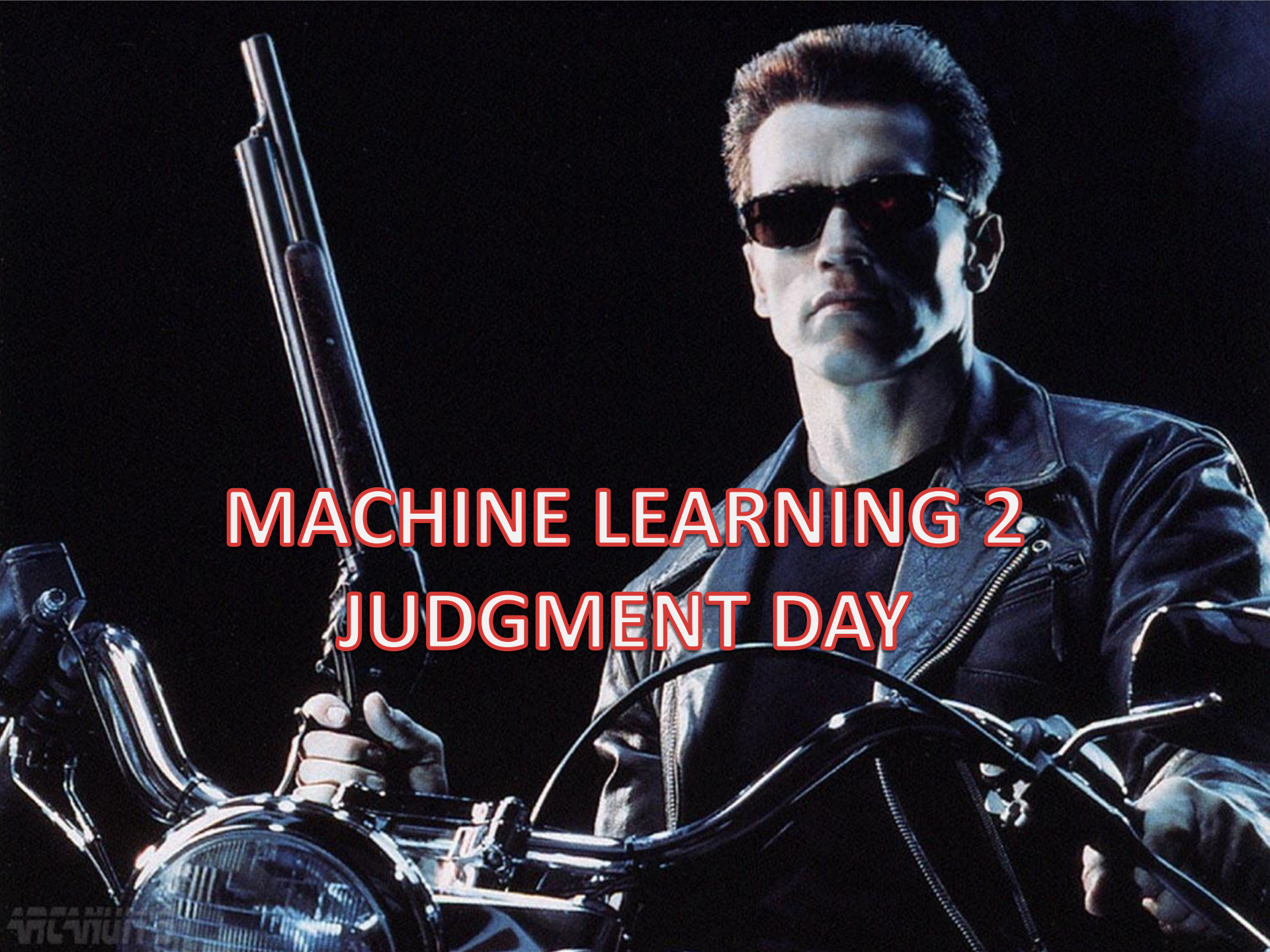

MACHINE LEARNING 2
JUDGMENT DAY

Reading “Quiz”

[redacted]

A Metric for Software Readability

Raymond P.L. Buse and Westley R. Weimer
Department of Computer Science
University of Virginia
Charlottesville, VA, USA
{buse, weimer}@cs.virginia.edu

ABSTRACT

In this paper, we explore the concept of code readability and investigate its relation to software quality. With data collected from human annotators, we derive associations between a simple set of local code features and human notions of readability. Using those features, we construct an automated readability measure and show that it can be 80% effective, and better than a human on average, at predicting readability judgments. Furthermore, we show that this metric correlates strongly with two traditional measures of software quality, code changes and defect reports. Finally, we discuss the implications of this study on programming language design and engineering practice. For example, our data suggests that comments, in of themselves, are less important than simple blank lines to local judgments of readability.

Categories and Subject Descriptors

D.2.9 [**Management**]: Software quality assurance (SQA); D.2.8 [**Software Engineering**]: Metrics

General Terms

a project [1]. Other researchers have noted that the act of reading code is the most time-consuming component of all maintenance activities [29, 36, 38]. Furthermore, maintaining software often means evolving software, and modifying existing code is a large part of modern software engineering [35]. Readability is so significant, in fact, that Elshoff and Marcotty proposed adding a development phase in which the program is made more readable [11]. Knight and Myers suggested that one phase of software inspection should be a check of the source code for readability [26]. Hanoef proposed the addition of a dedicated readability and documentation group to the development team [19].

We hypothesize that everyone who has written code has some intuitive notion of this concept, and that program features such indentation (e.g., as in Python [43]), choice of identifier names [37], and comments are likely to play a part. Dijkstra, for example, claimed that the readability of a program depends largely upon the simplicity of its sequencing control, and employed that notion to help motivate his top-down approach to system design [10].

We present a descriptive model of software readability based on simple features that can be extracted automatically from programs. This model of software readability

Review

- Often in PL we try to form judgments about complex human-related phenomena
- The right answer arises from a complex interaction of features
- ML can help us model this sort of thing

Examples

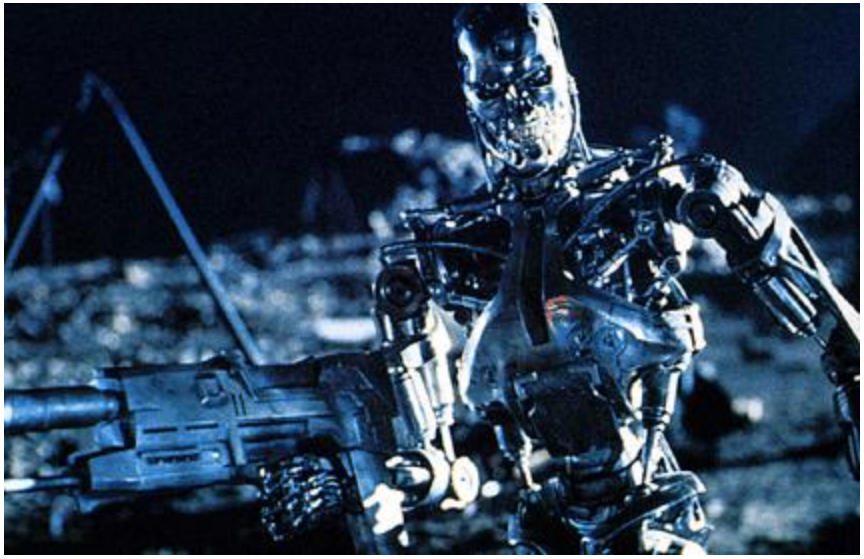
- Bug Reports
- Readability
- Documentation
- Program Synthesis
- Genetic Programming

Last Time

- ML Basics
- ML (heart) PL
- K-means Clustering (Unsupervised)
- Linear Regression (Supervised)

This Exciting Episode

- Quick review of basics (now with formalisms)
- “Advanced” ML Algorithms (supervised only)
- Coercing a problem in ML
- Evaluation Techniques
- Other “useful” info



Review: Supervised Learning

Given a set of example pairs (instances, answers)

$$\mathrm{T} = \{(\vec{x}, y) \,|\, \vec{x} \in X, y \in Y\}$$

Find a function (in the allowed class)

$$f : X \to Y, f \in F$$

That minimizes some cost function

$$C : \langle F, \mathrm{T} \rangle \to \Re$$

Example: Linear Regression

$$f(\vec{x}) = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \ldots + \beta_n \cdot x_n$$

$$C = \frac{1}{|\mathrm{T}|} \sum_{i=1}^{|\mathrm{T}|} (f(x_i) - y_i)^2$$

Example: Logistic Regression

$$f(\vec{x}) = \frac{1}{1+e^{-z}}$$

$$z = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \ldots + \beta_n \cdot x_n$$

$$C = \frac{1}{|\mathrm{T}|} \sum_{i=1}^{|\mathrm{T}|} (f(x_i) - y_i)^2$$

Learning Goals

Numerical (scalar values):

- 3.456, 3.457, 12.3452

Ordinal (categories with an ordering):

- Low, Medium, High

Nominal (categories with no ordering):

- Tree, Car, Fire

Advanced ML Algorithms

- Support Vector Machines
- Bayesian
- Decision Trees
- Artificial Neural Networks

Support Vector Machines

Support Vector Machines

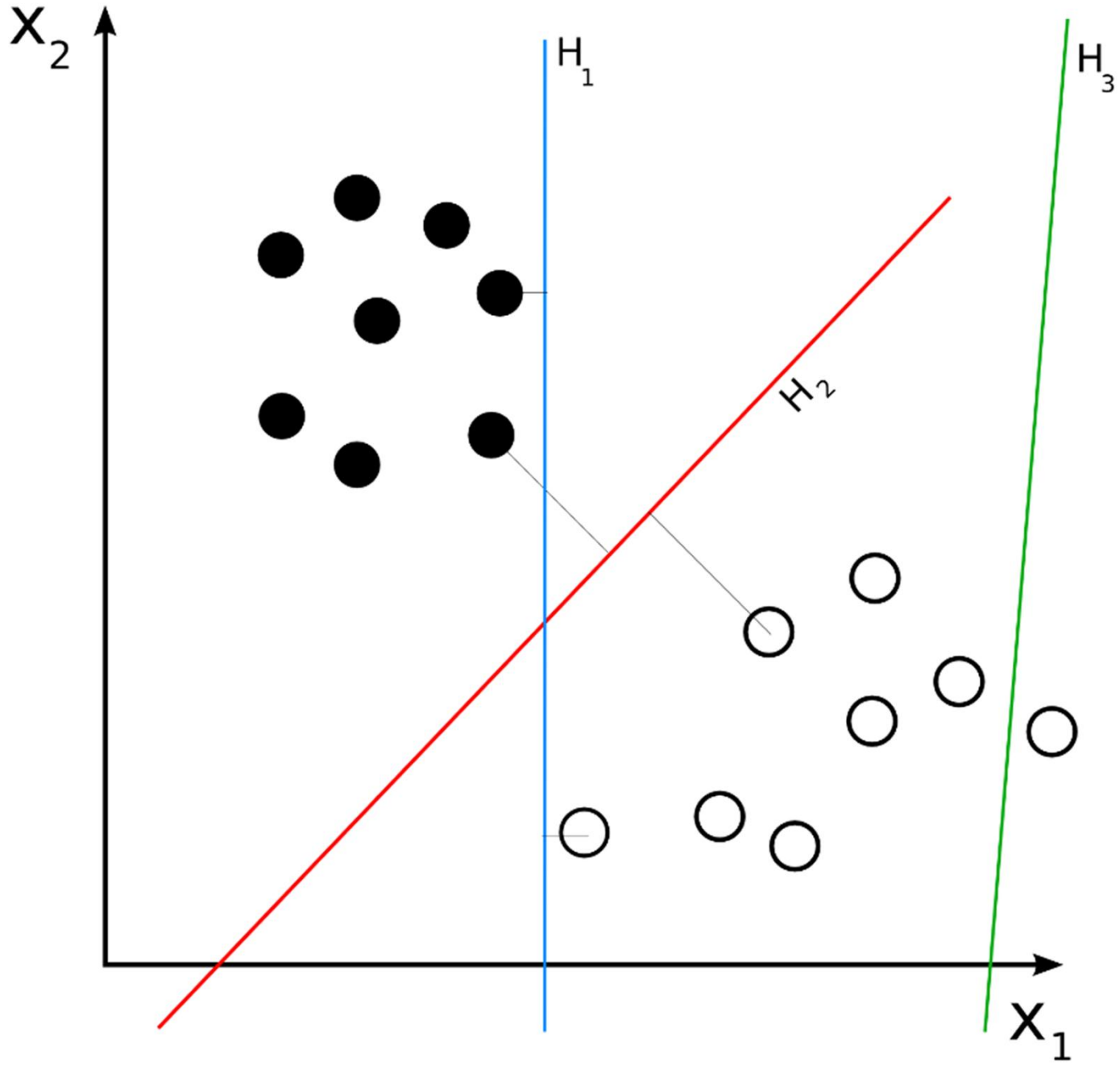
Generalized Linear Classifiers

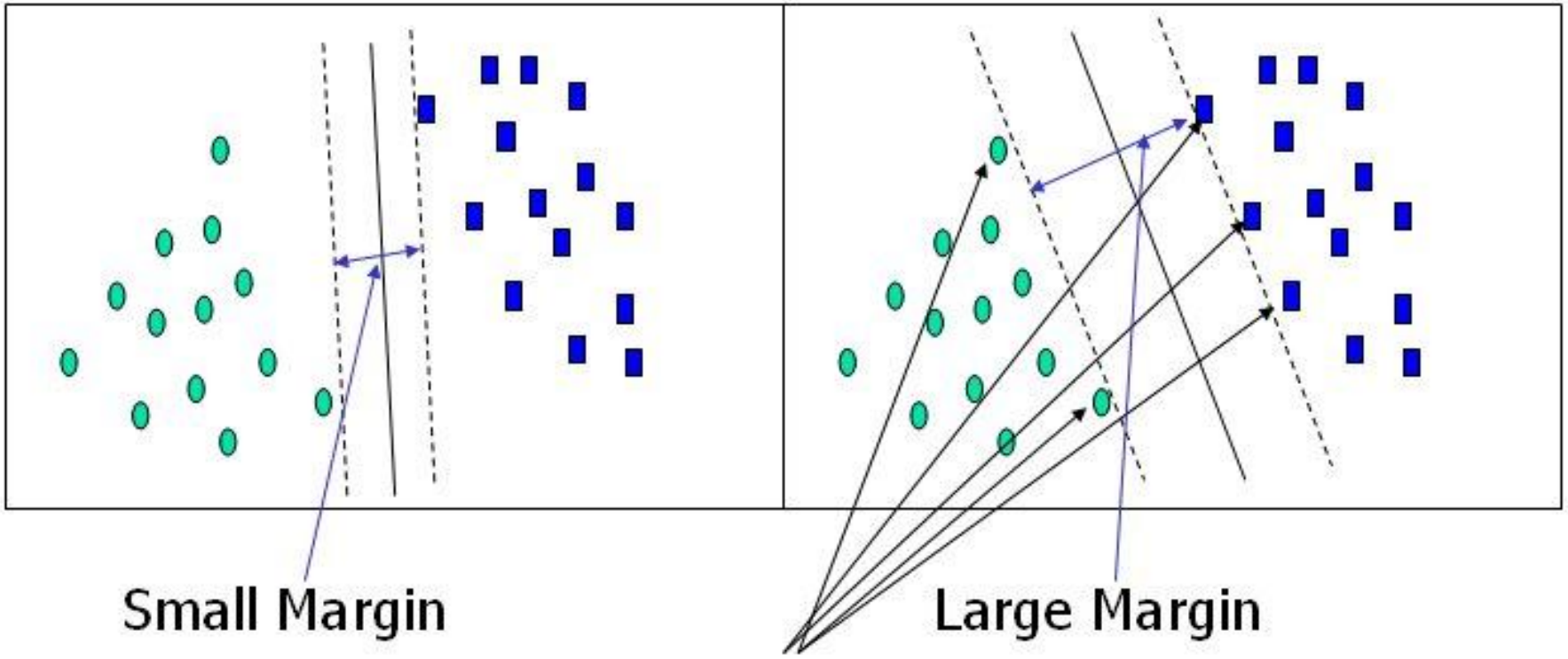
Binary Class Version:

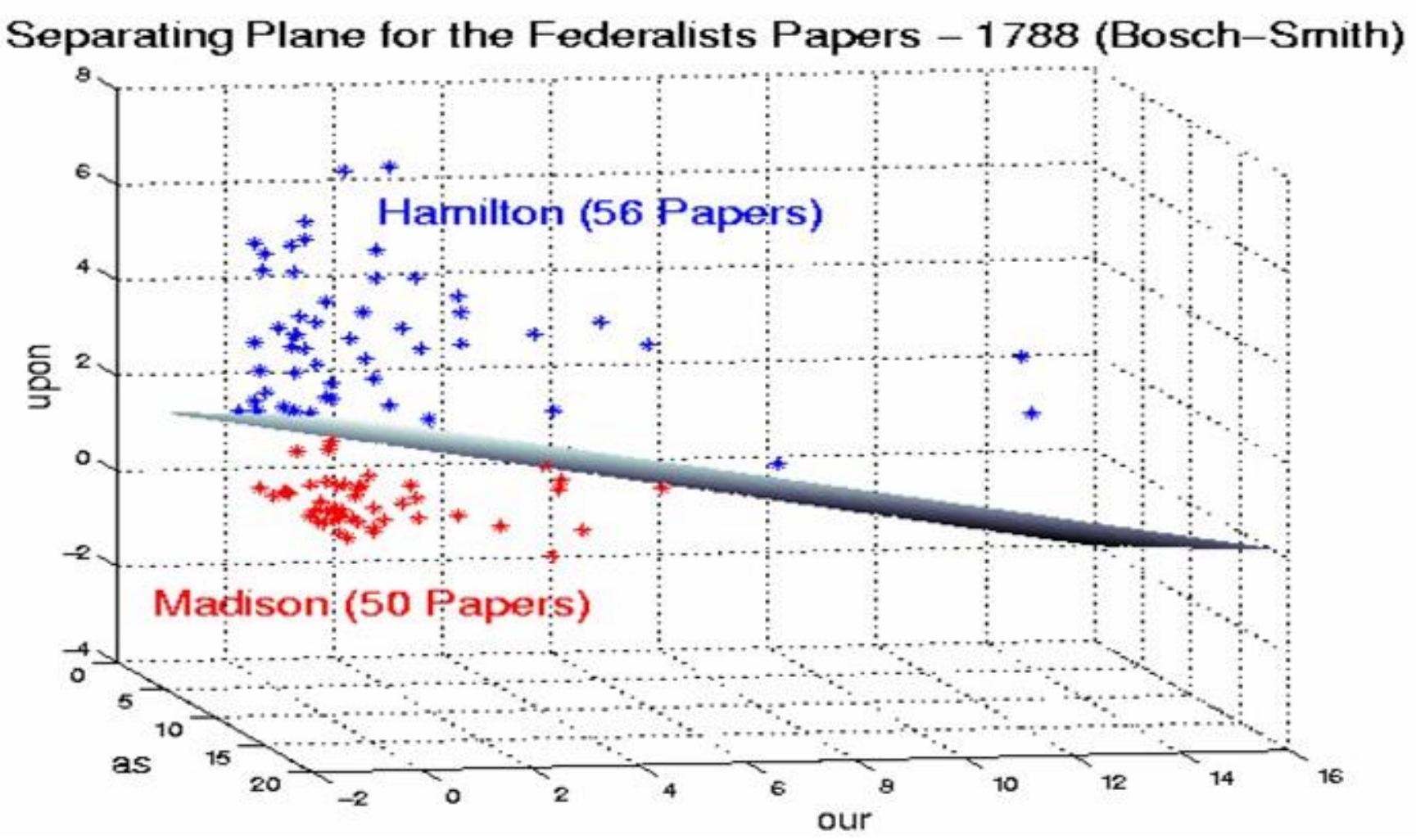
- Given Training Data

$$\mathrm{T} = \{(\vec{x}, y) \,|\, \vec{x} \in \Re^p, y \in \{-1,1\}\}$$

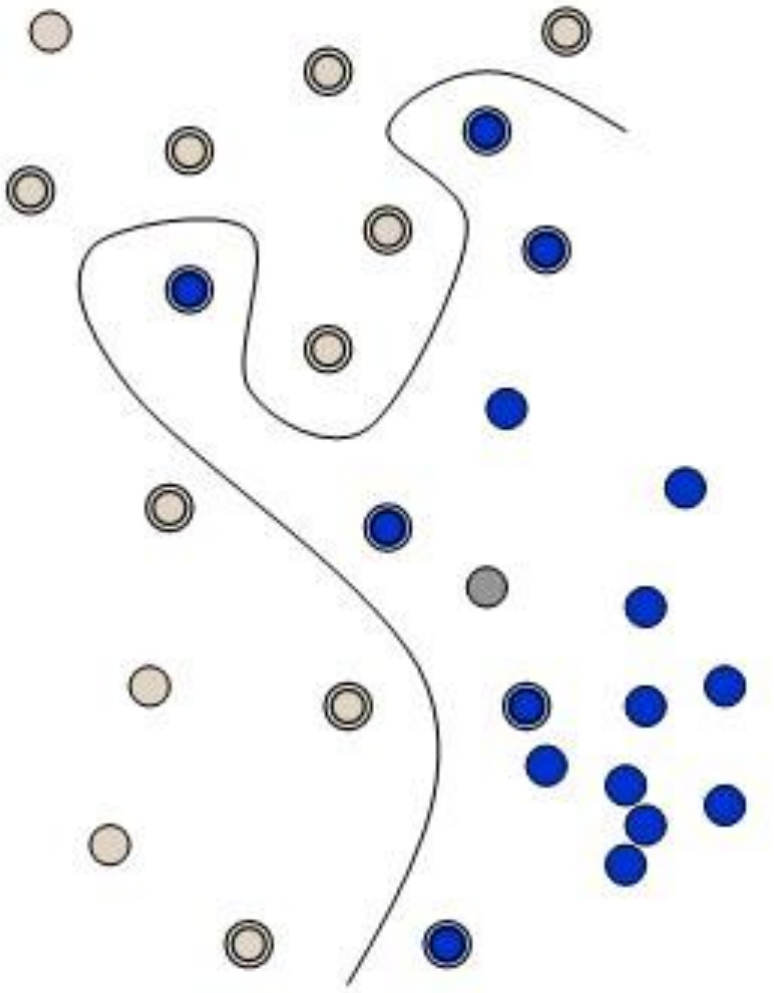
- We want to give the *hyperplane* that divides the points having y=-1 from y=1 that has "maximal margin"


X2
H1
H3
H2
X1


Small Margin
Large Margin
Support Vectors

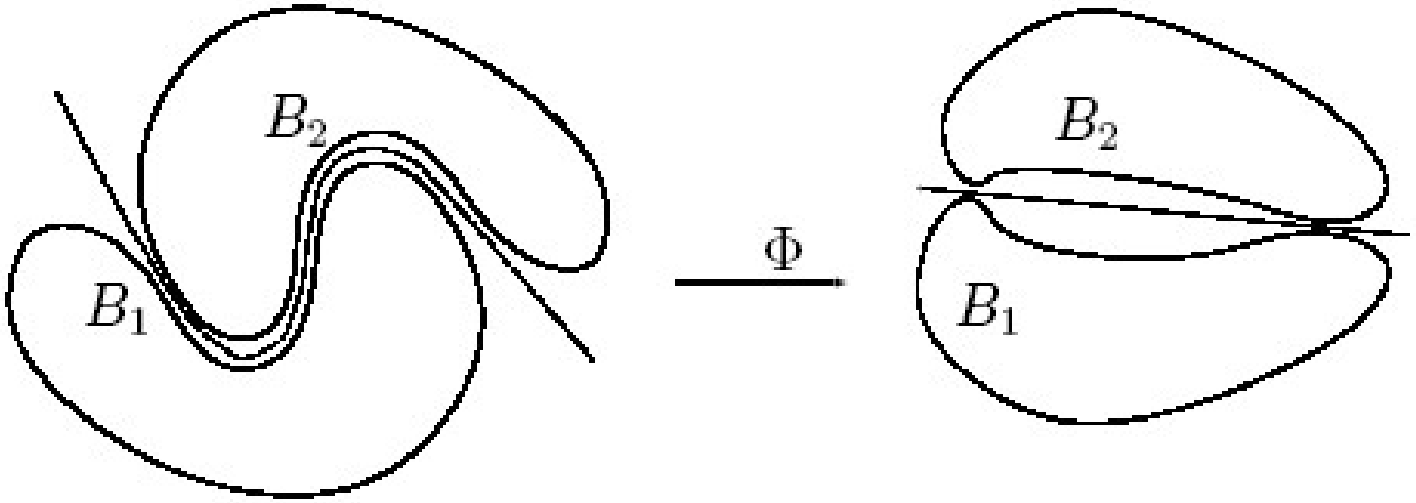

Separating Plane for the Federalists Papers – 1788 (Bosch–Smith)
Hamilton (56 Papers)
Madison (50 Papers)
upon
as
our

When Straight Lines go Crooked



When Straight Lines go Crooked

- Rather than fitting nonlinear curves to the data, SVM handles this by using a *kernel function* to map the data into a different space

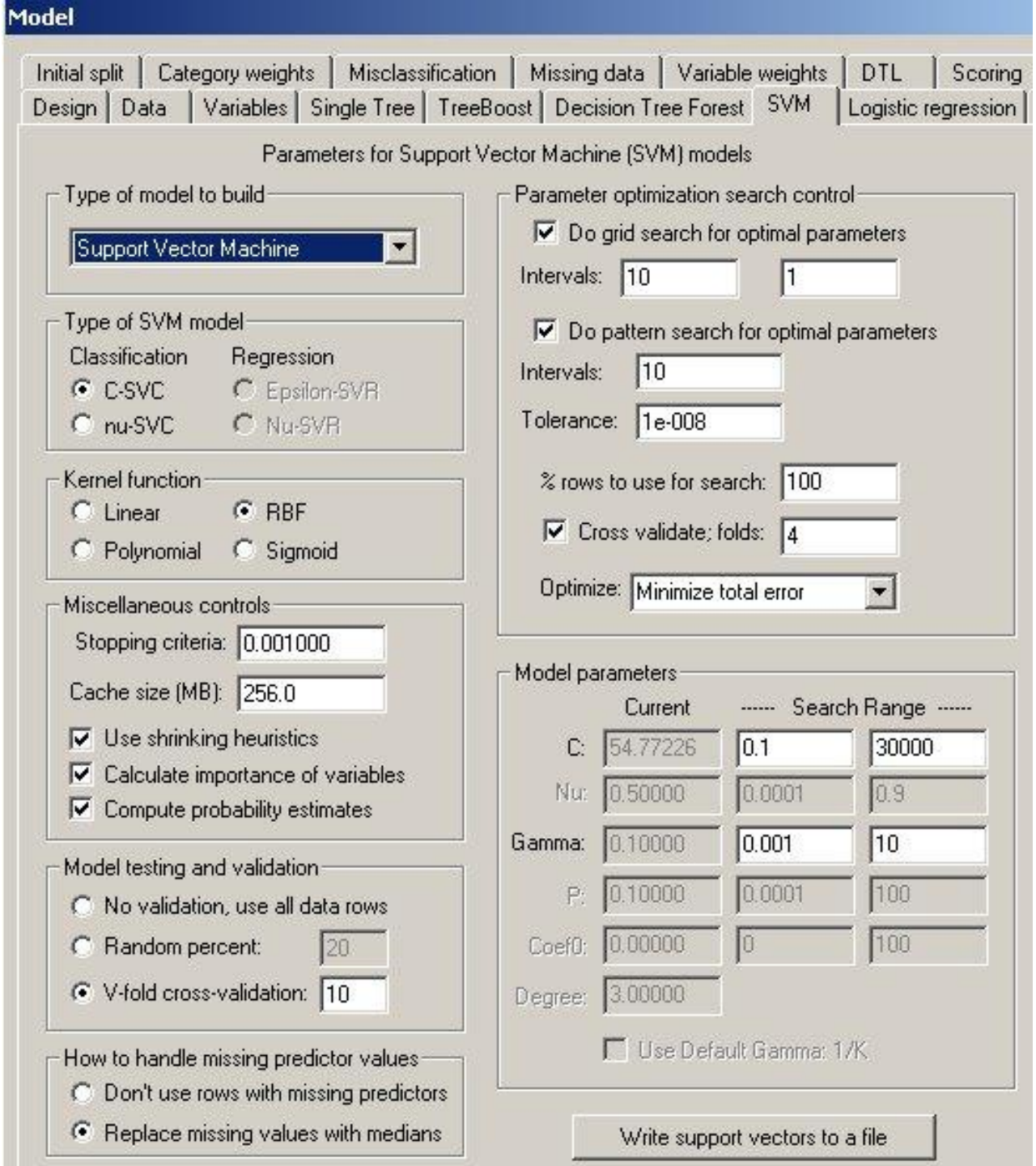


When Perfect Separation is not possible

- SVM models have a cost parameter, *C*, that controls the trade off between allowing training errors and forcing rigid margins.

Non-binary SVM

- *one against many* - k models where each class is split out and all of the other classes are merged
- *one against one* - $k(k-1)/2$ models - all pair wise combinations.


Model
Initial split
Category weights
Misclassification
Missing data
Variable weights
DTL
Scoring
Design
Data
Variables
Single Tree
TreeBoost
Decision Tree Forest
SVM
Logistic regression
Parameters for Support Vector Machine (SVM) models
Type of model to build
Support Vector Machine
Type of SVM model
Classification
Regression
C-SVC
Epsilon-SVR
nu-SVC
Nu-SVR
Kernel function
Linear
RBF
Polynomial
Sigmoid
Miscellaneous controls
Stopping criteria: 0.001000
Cache size (MB): 256.0
Use shrinking heuristics
Calculate importance of variables
Compute probability estimates
Model testing and validation
No validation, use all data rows
Random percent: 20
V-fold cross-validation: 10
How to handle missing predictor values
Don't use rows with missing predictors
Replace missing values with medians
Parameter optimization search control
Do grid search for optimal parameters
Intervals: 10 1
Do pattern search for optimal parameters
Intervals: 10
Tolerance: 1e-008
% rows to use for search: 100
Cross validate; folds: 4
Optimize: Minimize total error
Model parameters
Current
Search Range
C: 54.77226 0.1 30000
Nu: 0.50000 0.0001 0.9
Gamma: 0.10000 0.001 10
P: 0.10000 0.0001 100
Coef0: 0.00000 0 100
Degree: 3.00000
Use Default Gamma: 1/K
Write support vectors to a file

SVM Tradeoffs

Advantages

- Hypothesis has an explicit dependence on data (support vectors)
- Few tuning parameters

Disadvantages

- No clear estimate of “confidence” to go along with classification. Next: A probabilistic model…

Bayesian Learner

A Probabilistic Classifier

- What's the probability of class C, given feature vector x?

$$P(C \mid x_{1,} \ldots x_n)$$

- The problem: If features can take on r different values, there are r^n different probabilities to consider!
- Bayes' Theorem to the rescue…

Bayes' Theorem

- Bayes' theorem describes the way in which one's beliefs about observing 'A' are updated by having observed 'B'.

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Bayesian Example

- Suppose 6610 has 90% civilians and 10% terminators as students.
  - About half of terminators wear sunglasses.
  - None of the civilians wear sunglasses.
- You see a (random) student who is NOT wearing sunglasses. What is the probability this student is a terminator?

Bayesian Example

- Call event *A* the student is a terminator and the event *B* that the student is NOT wearing sunglasses.

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} = \frac{0.5 \cdot 0.1}{0.9} = 0.055$$

Applying Bayes' Theorem to ML

$$P(C \mid x_{1,} \ldots x_n) = \frac{P(C)P(x_{1,} \ldots x_n \mid C)}{P(x_{1,} \ldots x_n)}$$

- Denominator is effectively constant.
- Assuming conditional independence

$$P(x_i, x_j \mid C) = P(x_i \mid C)P(x_j \mid C)$$

- We can say:

$$P(C \mid x_{1,} \ldots x_n) = P(C)\prod_{i=1}^{n} P(x_i \mid C)$$

The Bayesian Classifier

$$P(C \mid x_{1,} \ldots x_n) = P(C)\prod_{i=1}^{n} P(x_i \mid C)$$

- With k classes, n features, and r values per feature there are now

  k + n·k·r probabilities

Bayesian Advantages

- Fast to train, fast to classify
- Well suited for nominal features
- Good for combining models:
  - Suppose we have two kinds of feature vectors, x_1 and x_2 Rather than building a monolithic model, it might be more practical to learn two separate classifiers, $P(C|x_1)$ and $P(C|x_2)$ and then to combine them.

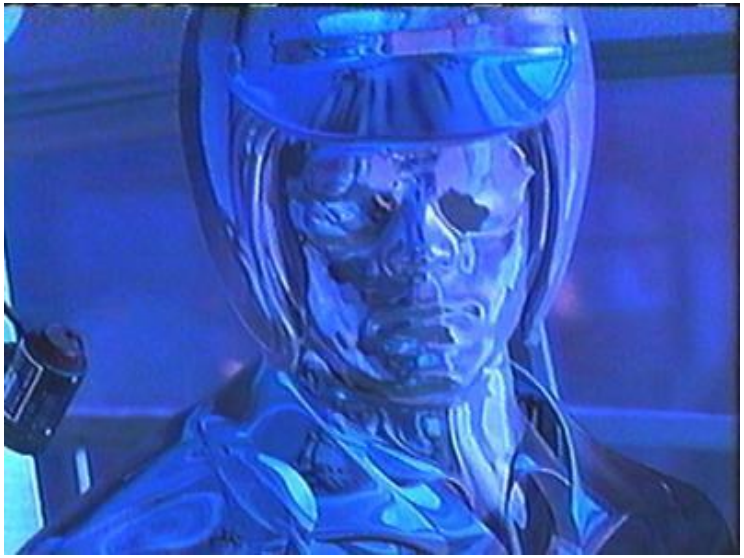
$$P(C \mid x_1, x_2) = \frac{P(C \mid x_1)P(C \mid x_2)}{P(C)}$$

Bayesian Disadvantages

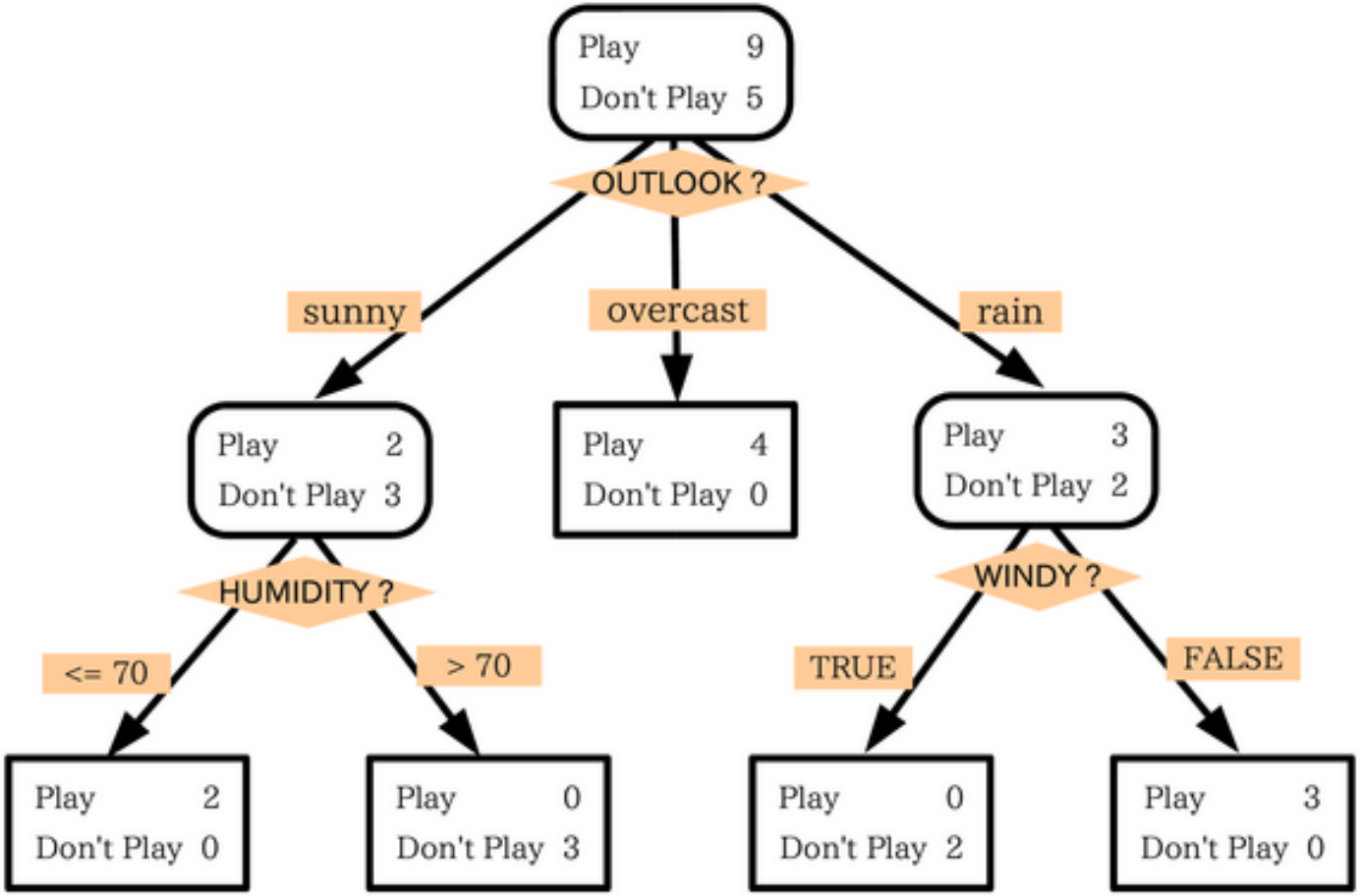
- If conditional independence isn't true (at least most of the time), it may not make for a very good model

Advanced ML Algorithms

- Support Vector Machines
- Bayesian
- Decision Trees
- Artificial Neural Networks



Decision Trees



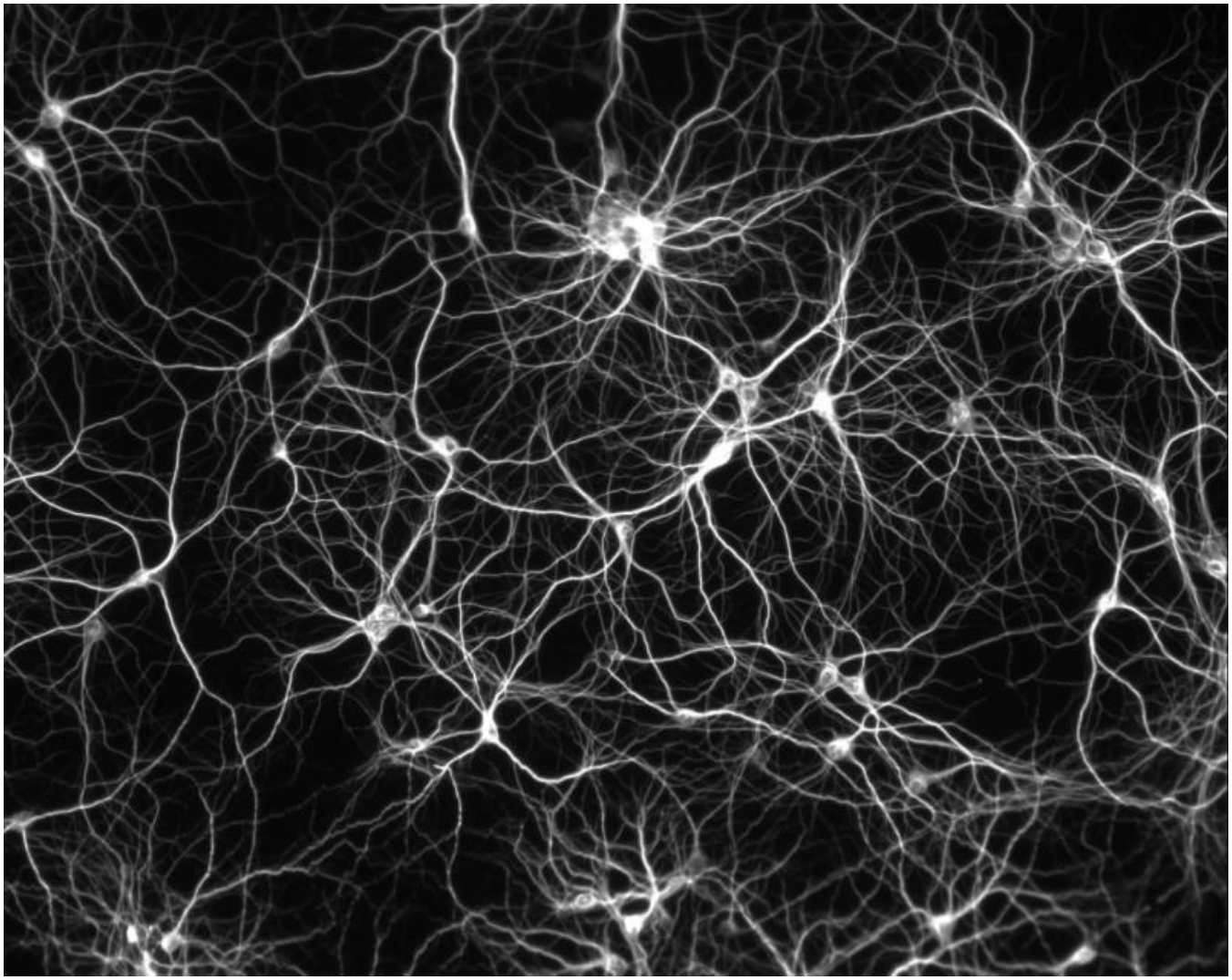
Automatic Decision Tree Building

- Most algorithms are variations on a top-down, greedy search through the space of possible decision trees.
- Search through the attributes of the training instances and pick the attribute that best separates the given examples.
- If the attribute perfectly classifies the training sets then stop.
- Otherwise recursively operate on the partitioned subsets to get their "best" attribute.

Decision Tree Advantages

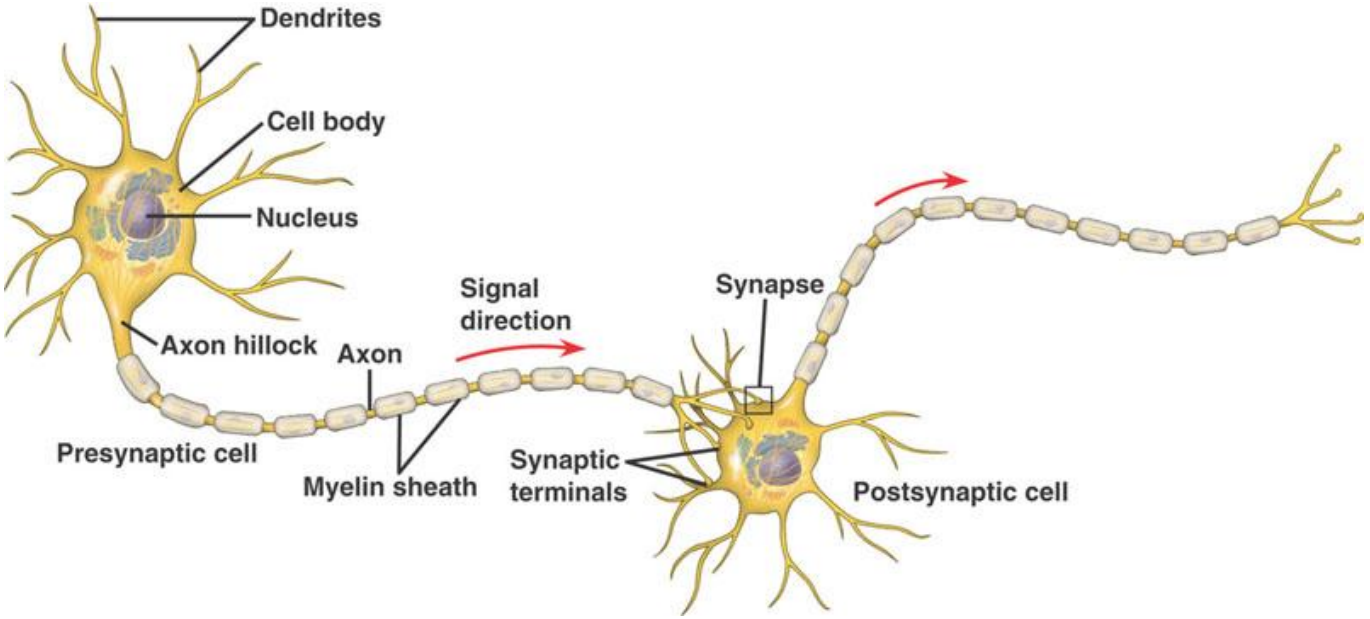
- Simple to understand and interpret.
- Able to handle both numerical and categorical data.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Decisions are fast

Artificial Neural Networks



ANNs

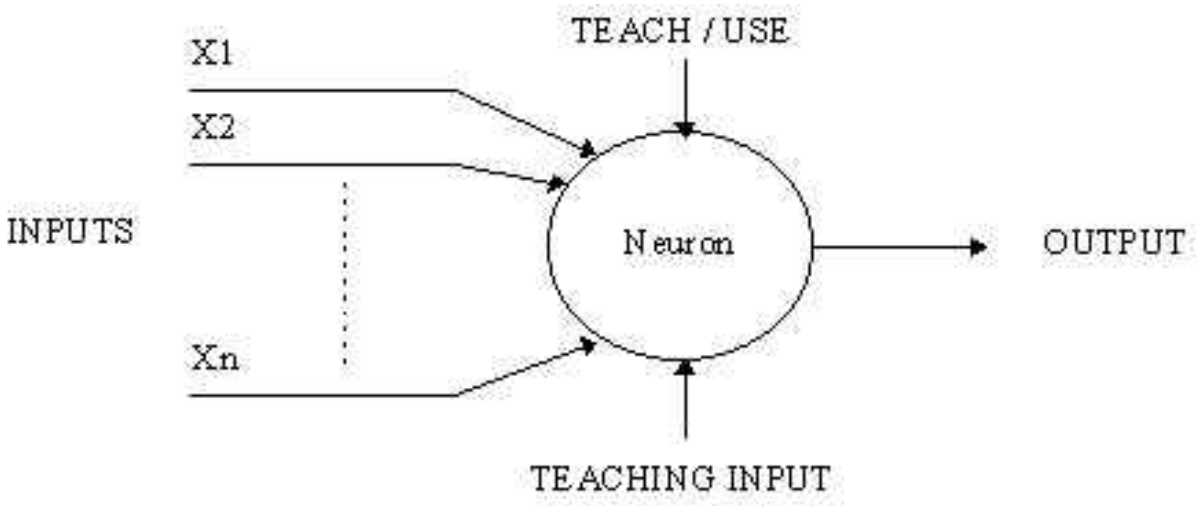
- Idea: simulate biological neural networks



- Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes.

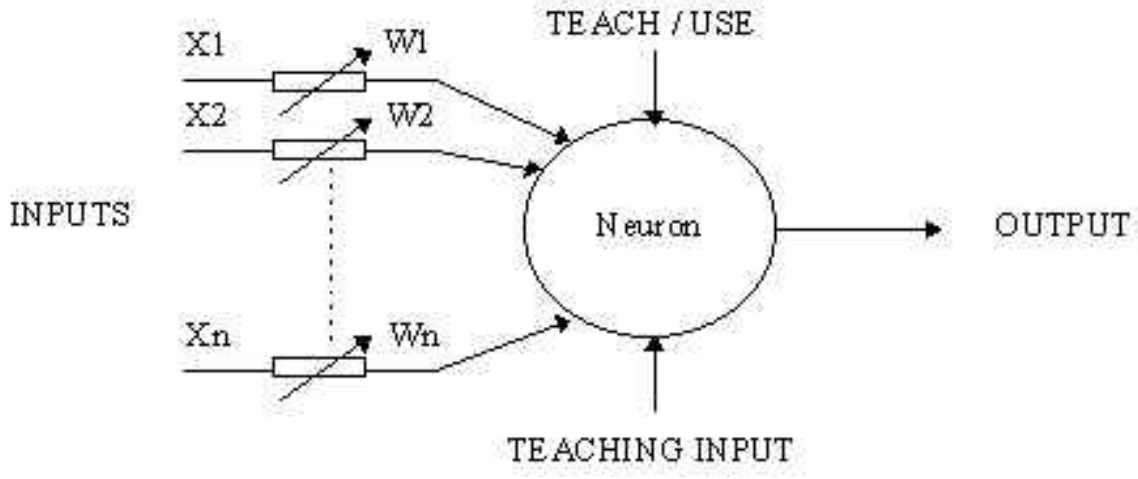
A Simple Firing Rule

- Take a collection of training patterns for a node.
- Patterns not in the collection cause the node to fire if, on comparison, they have more input elements in common with the 'nearest' pattern in the firing set than with the 'nearest' pattern in the non-firing set.

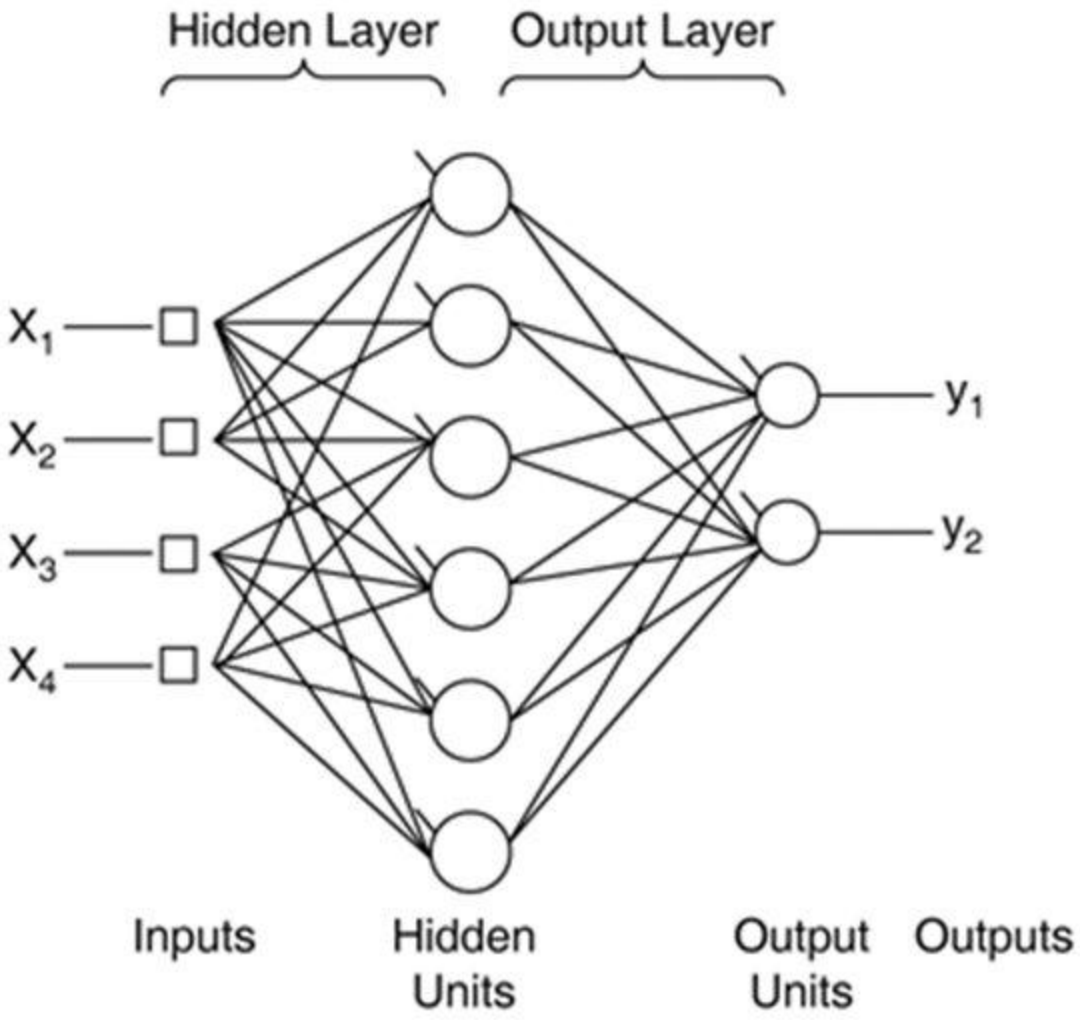


A More Complicated Neuron

- Weight the inputs.
- These weighted inputs are then added together and if they exceed a pre-set threshold value, the neuron fires. In any other case the neuron does not fire.



Multilayer Neural Net



ANN Success Stories

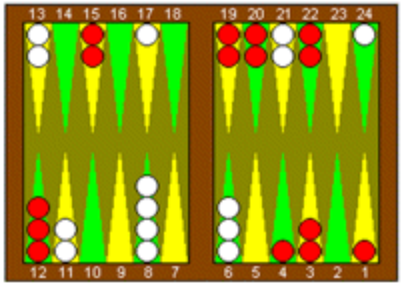
- Handwriting recognition
- Recognizing spoken words
- Face recognition
- ALVINN
- TD-BACKGAMMON

ALVINN

- Autonomous Land Vehicle in a Neural Network

- 1995
  - Drove 1000 miles in traffic at speed of up to 120 MPH
  - Steered the car coast to coast (throttle and brakes controlled by human)
- 30 x 32 image as input, 4 hidden units, and 30 outputs

TD-GAMMON



- Plays backgammon
- Created by Gerry Tesauro in the early 90s
- Uses variation of Q-learning
  - Neural network was used to learn the evaluation function
- Trained on over 1 million games played against itself
- Plays competitively at world class level

ANN Advantages

- Online learning
- Large dataset applications
- Their simple implementation and the existence of mostly local dependencies exhibited in the structure allows for fast, parallel implementations in hardware.

ANN Disadvantages

- Model is opaque
- Can take a long time to train
- Have to re-train whole thing when new features are added

You must choose…

How would you design a …

1. Spam Filter
2. Alarm system for a nuclear power plant
3. Terminator (Learn to act like a human, terminate John Connor)

Boosting

Can a set of **weak learners** create a single **strong learner**?

- Weak learner: classifier which is only slightly correlated with the true classification.
- Strong learner: a classifier that is arbitrarily well-correlated with the true classification.
- Most boosting algorithms consist of iteratively weighting weak learners in some way that is usually related to the weak learner's accuracy.
- After a weak learner is added, the data is reweighted: examples that are misclassified gain weight and examples that are classified correctly lose weight.

Boosting & Cake Eating

- Can only work when the different learners have learned different things
- Combining is non-trivial
- Boosted decision trees a probably most popular

Now: Practical Stuff

- Features (find, selecting, etc.)
- Evaluating a model
- External Validity
- Other things to think about
- Where to go for code etc.

Features

Forming a model

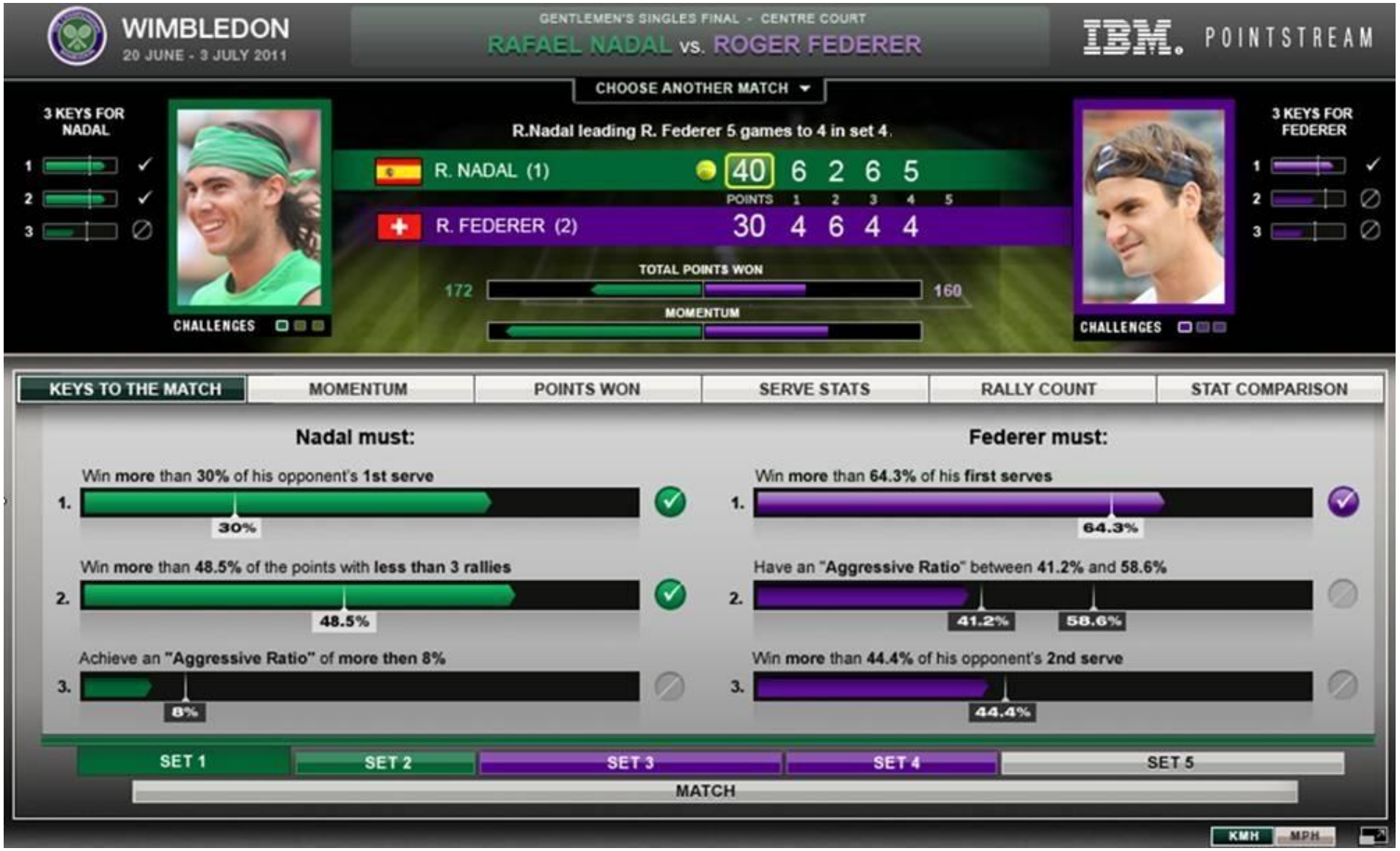
- ML is a recipe for solving a certain type of problem:

$$\mathrm{T} = \{(\vec{x}, y) \,|\, \vec{x} \in X, y \in Y\}$$

$$f : X \to Y, f \in F$$

$$C : \langle F, \mathrm{T} \rangle \to \Re$$

- Trick is to formulate a predictive model by choosing the right “features”


WIMBLEDON
20 JUNE - 3 JULY 2011
GENTLEMEN'S SINGLES FINAL - CENTRE COURT
RAFAEL NADAL vs. ROGER FEDERER
IBM. POINTSTREAM
CHOOSE ANOTHER MATCH
3 KEYS FOR NADAL
1
2
3
CHALLENGES
R.Nadal leading R. Federer 5 games to 4 in set 4.
R. NADAL (1) 40 6 2 6 5
POINTS 1 2 3 4 5
R. FEDERER (2) 30 4 6 4 4
TOTAL POINTS WON
172
160
MOMENTUM
CHALLENGES
3 KEYS FOR FEDERER
1
2
3
KEYS TO THE MATCH
MOMENTUM
POINTS WON
SERVE STATS
RALLY COUNT
STAT COMPARISON
Nadal must:
1. Win more than 30% of his opponent's 1st serve
30%
2. Win more than 48.5% of the points with less than 3 rallies
48.5%
3. Achieve an "Aggressive Ratio" of more then 8%
8%
Federer must:
1. Win more than 64.3% of his first serves
64.3%
2. Have an "Aggressive Ratio" between 41.2% and 58.6%
41.2%
58.6%
3. Win more than 44.4% of his opponent's 2nd serve
44.4%
SET 1
SET 2
SET 3
SET 4
SET 5
MATCH
KMH
MPH

Features

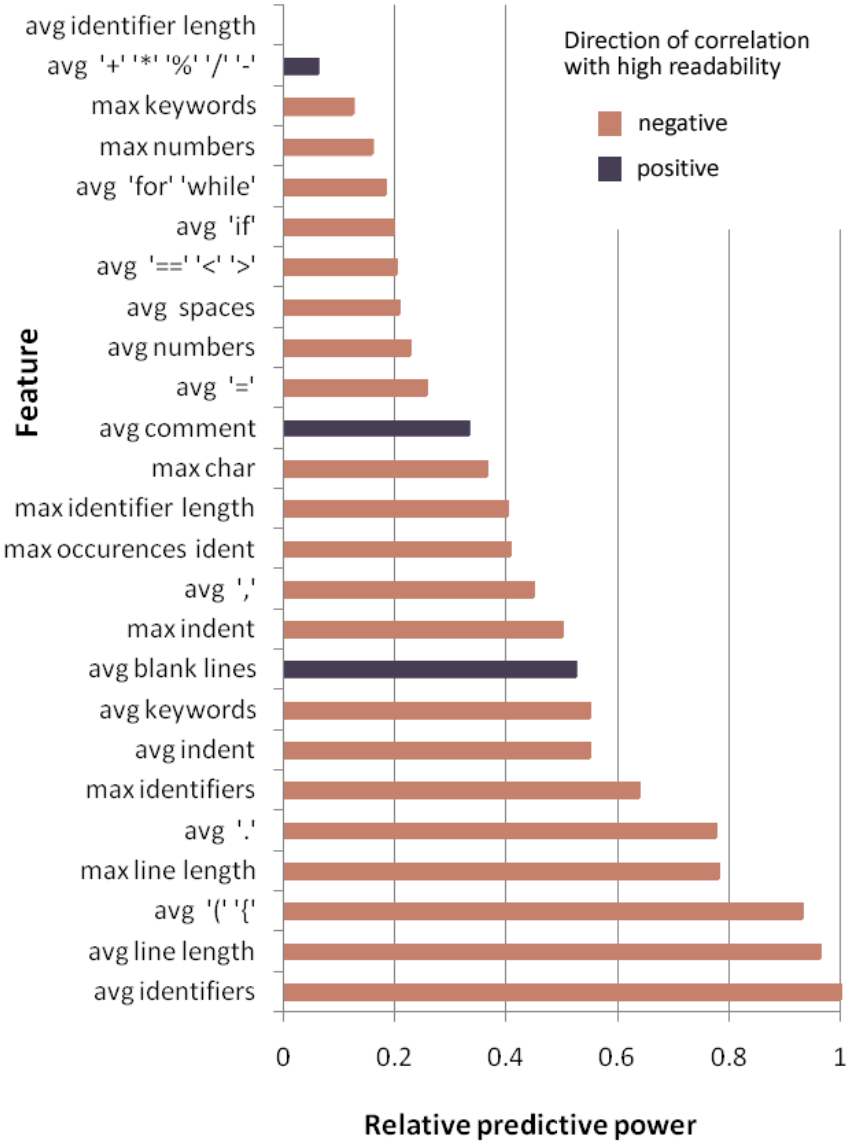
```
/**
 * Moves this unit to america.
 *
 * @exception IllegalStateException If the
move is illegal.
 */
public void moveToAmerica() {

   if (!(getLocation() instanceof Europe))
{
      throw new IllegalStateException("A
unit can only be "
         + "moved to america from
europe.");
   }

   setState(TO_AMERICA);

   // Clear the alreadyOnHighSea flag
   alreadyOnHighSea = false;
}
```

feature	value
Comments	2
Function calls	3
LOC	19
Has Exception?	TRUE

Feature Importance

- Often we want to understand which features are most predictive
- Three Basic Techniques
  - Inspect Model Directly (Not always possible)
  - Leave-one-out analysis (Train on all features but one)
  - Singleton feature analysis (Train on each feature individually)

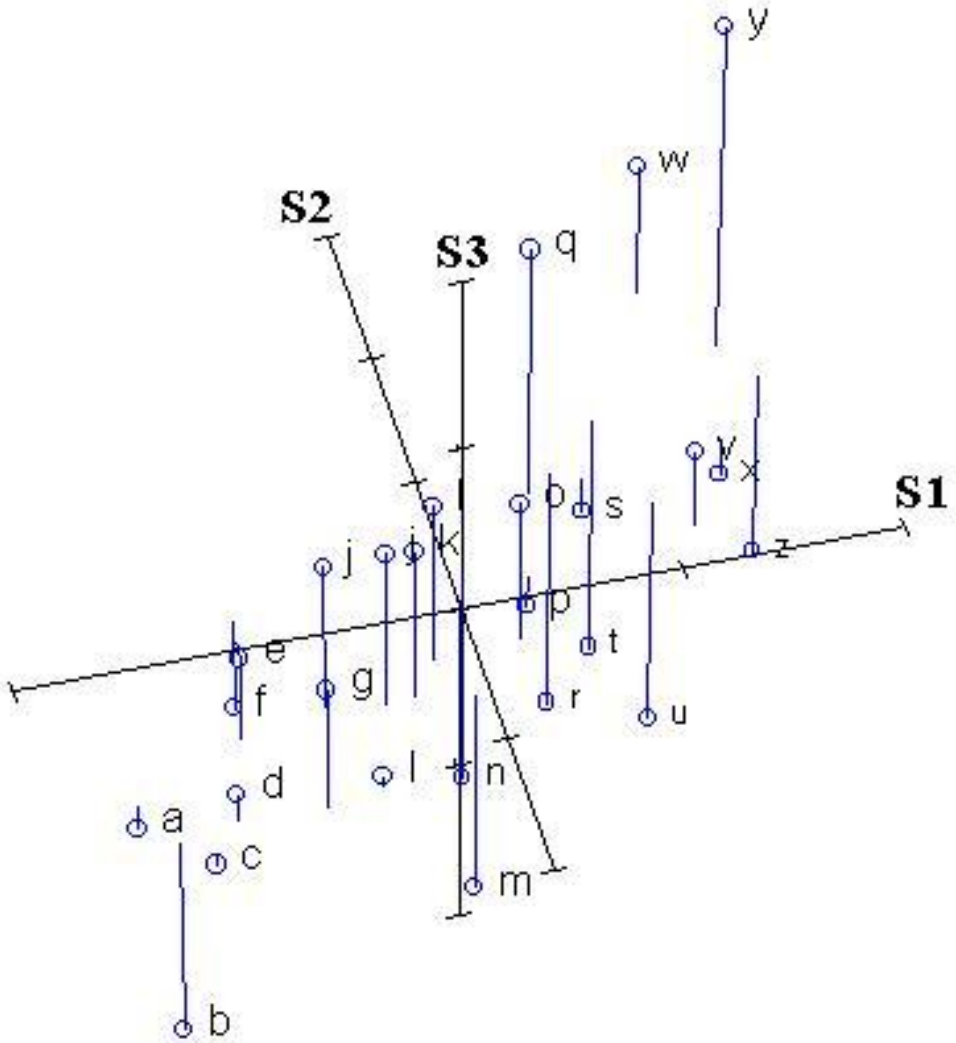
Feature Importance Example



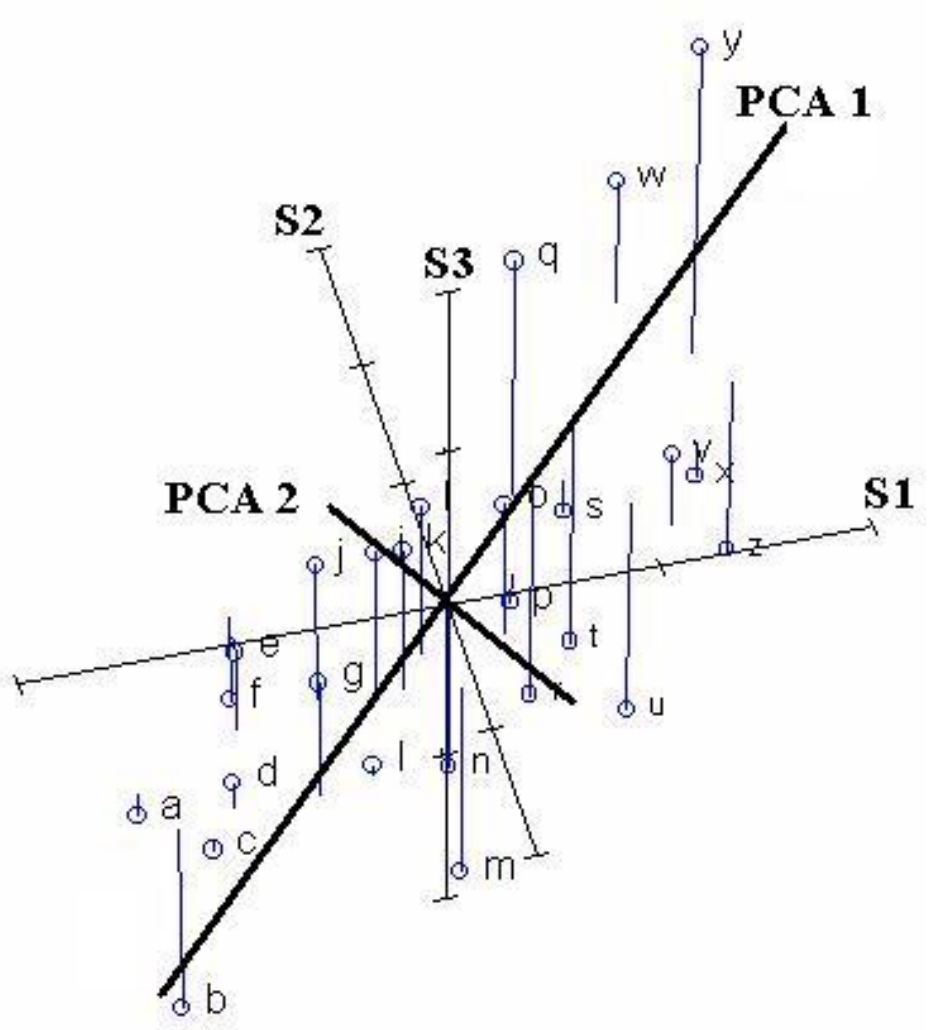
When Features Overlap

- Sometimes we want to know how many features are actually needed
- Example:
  - Feature 1: Lines of code
  - Feature 2: $\text{Lines of code} \cdot 2$
- Solution: Principle Component Analysis (PCA)
- Idea: Iteratively perform linear regressions, find out how many vectors (ideal features) are needed to account for variance in the data.

PCA



PCA



PCA

- Each PC reduces dimensionality such that the total linear variability is reduced
- Repeat until no dimensions left
- Can only account for *linear* variability

Curse of Dimensionality

- More features is NOT always “more better”
- Problems tend to become intractable as number of variables increases
- Data spreads out exponentially with the number of dimensions (makes data sparse in high dimensions)
- Makes any kind of attempt to search, sort, filter or organize nearly impossible as you will have no way to compare two items in your data set with values along mostly disjoint sets of variables.

Feature Selection

- Alleviating *the curse of dimensionality*
- Enhances generalization capability
- Speeds up learning process
- Improves model interpretability

Feature Selection

- Optimal feature selection requires an exhaustive search of all possible subsets of features of size N
- Feature ranking: rank the features by a metric and choose the top N
- Stepwise Regression: Greedy algorithm that adds the best feature (or deletes the worst feature) at each round until N are left.

Evaluation

Direct Model Evaluation

- How closely does the model conform to the data?
- This is non-trivial
- Major Issues
  - Bias
  - Statistical Significance
  - Over-fitting

Evaluation Strategy 1: Correlation

- Suppose we have a vector of correct answers X, and model output Y
- Question: How similar are X and Y?
- Pearson Product Moment Correlation Coefficient : r
- If r^2 is 0.90, then 90% of the variance of *Y* can be “explained" by the linear relationship between *X* and *Y*
- But what if the data is categorical?

Evaluation Strategy 2: f-measure

- You have a binary classifier for “will this report be resolved in <= 30 days”
- You have 27,984 reports with known answers
  - C = correct set of reports resolved in 30 days
  - R = set of reports the model returns

Precision = $\frac{|C \cap R|}{|R|}$ Recall = $\frac{|C \cap R|}{|C|}$

f-measure= $\frac{2 \cdot P \cdot R}{P + R}$

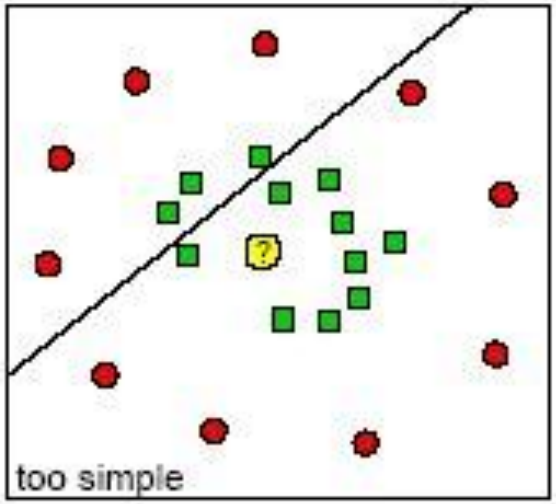
f-measure and Bias

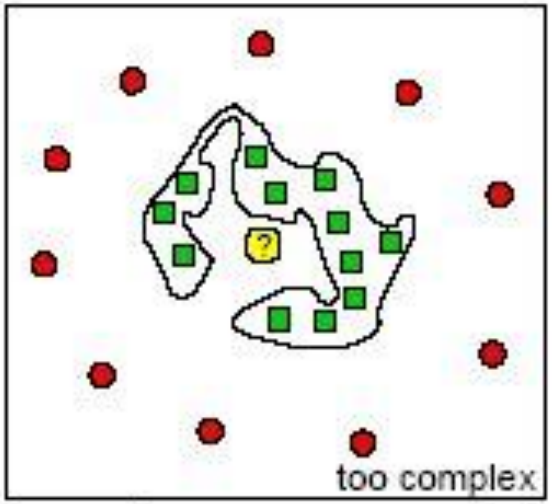
- Say you have 100 instances
- 50 yes instances, 50 no instances, at random
  - "Flip Fair Coin": Prec=0.5, Rec=0.5, F=0.5
  - "Always Guess Yes": Prec=0.5, Rec=1.0, F=0.66
- 70 yes instances, 30 no instances, at random
  - "Flip Fair Coin": Prec=0.7, Rec=0.5, F=0.58
  - "Flip Biased Coin": Prec=0.7, Rec=0.7, F=0.7
  - "Always Guess Yes": Prec=0.7, Rec=1.0, F=0.82
- May want to subsample to 50-50 split for evaluation purposes

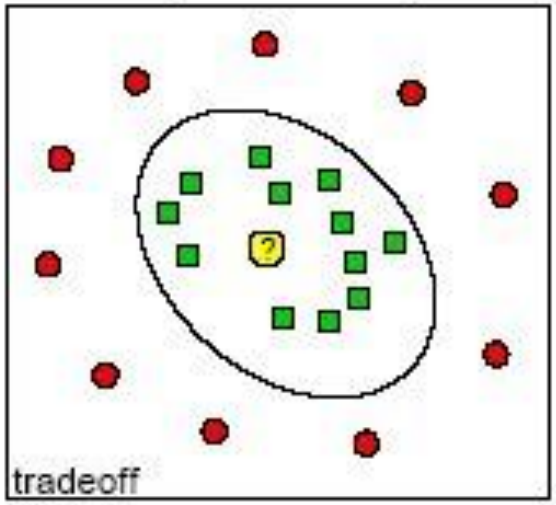
Other accuracy metrics

- % of instances correctly classified
- Cohen's Kappa
- Root mean squared error

Underfitting and Overfitting







negative example
positive example
new patient

Cross Validation to test for over-fitting

- Idea: Don't evaluate on the same data you trained on.

- N-Fold Cross-Validation
  - Partition instances into n subsets
  - Train on 2..n and test on 1
  - Train on 1, 3..n and test on 2, etc.

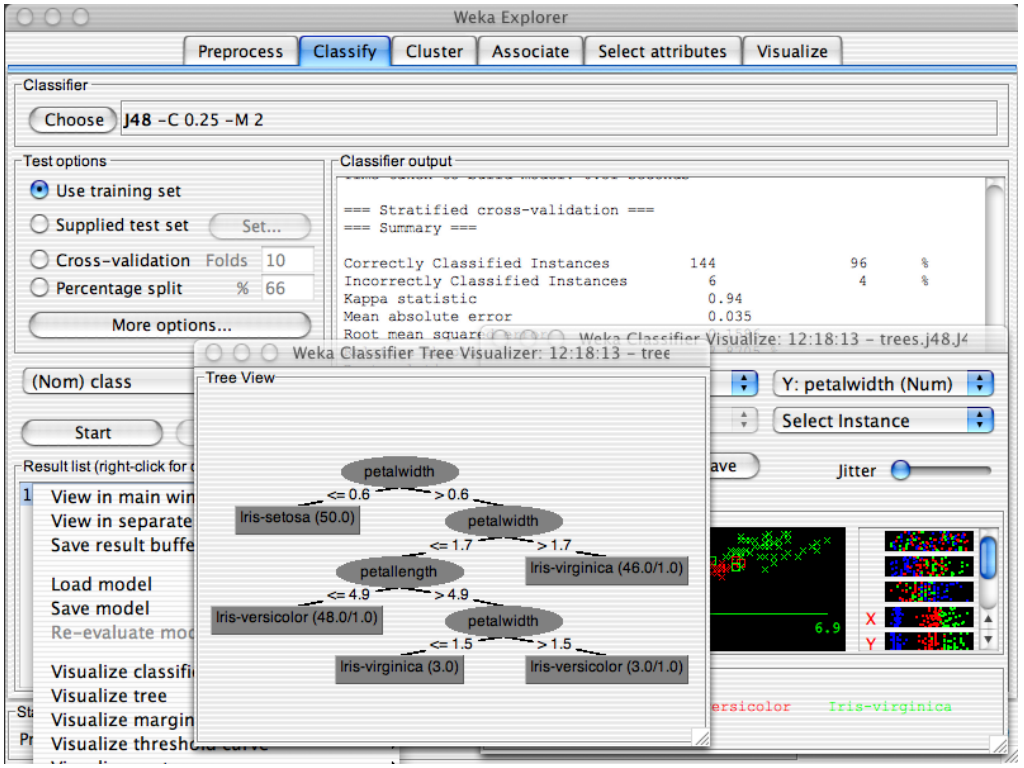
External Validity

- Sometimes we want to explore how well a model correlates with some external truth (e.g., to show utility)
- Solution:
  - Discrete Case: f-measure
  - Continuous Case: Pearson
  - Hybrid: Bins + Kendall's Tau

Even More Practical

ML implementations available in the environment of your choice

- Matlab
- R
- Weka (Java)
- Mathmatica
- [Many others]



The End?