Week 2

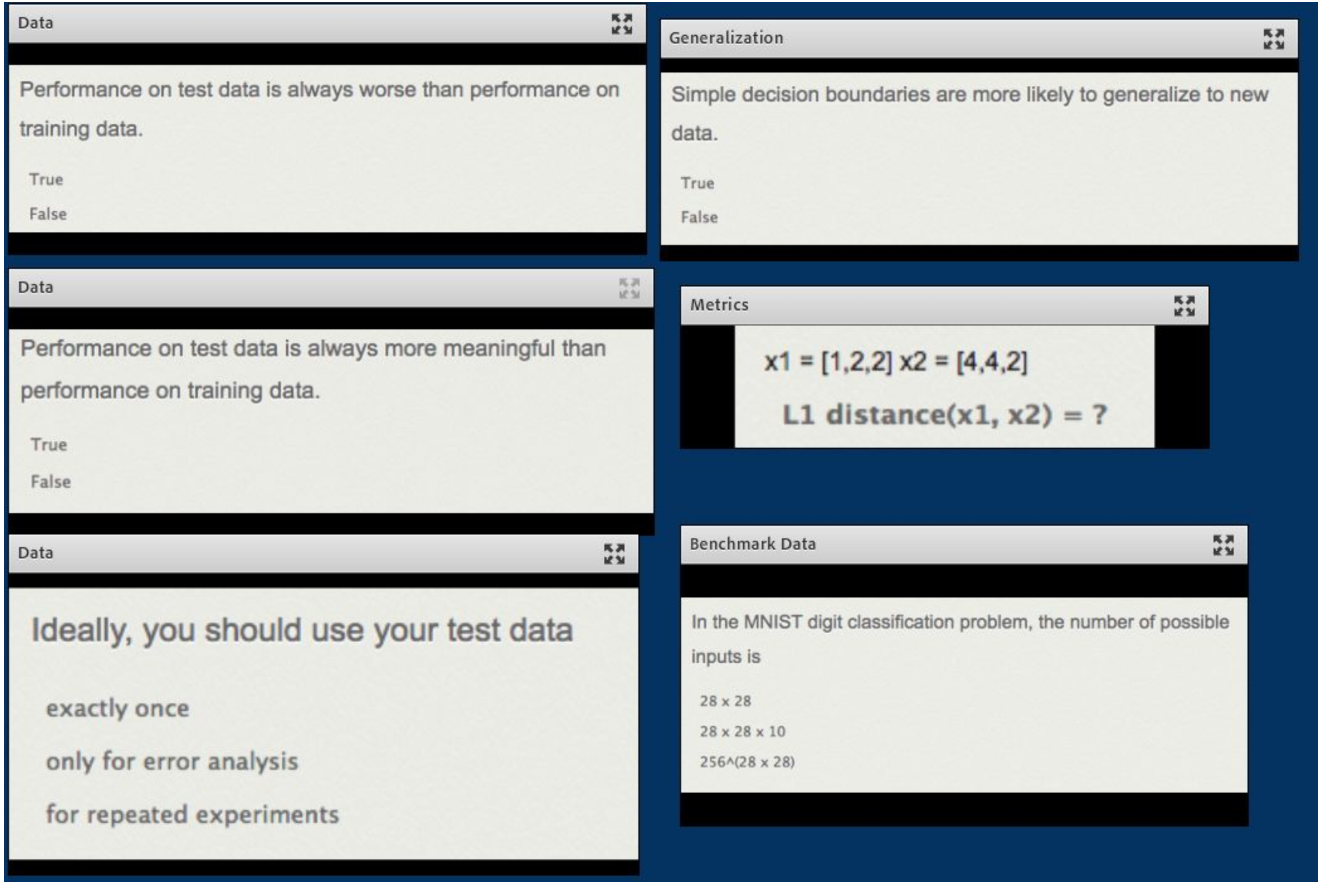
Agenda

1. Async Review
2. Parameters & Hyperparameters discussion
3. Similarity measure discussion
4. KNN + Applications
5. ML workflows discussion
6. Domingos paper discussion
7. Time permitting: KNN Notebooks

For next week:

- Read both Graham and Norvig articles

Warmup



Data

Performance on test data is always worse than performance on training data.

True

False

Data

Performance on test data is always more meaningful than performance on training data.

True

False

Data

Ideally, you should use your test data

exactly once

only for error analysis

for repeated experiments

Generalization

Simple decision boundaries are more likely to generalize to new data.

True

False

Metrics

x1 = [1,2,2] x2 = [4,4,2]

L1 distance(x1, x2) = ?

Benchmark Data

In the MNIST digit classification problem, the number of possible inputs is

28 x 28

28 x 28 x 10

256^(28 x 28)

Parameters and Hyperparameters

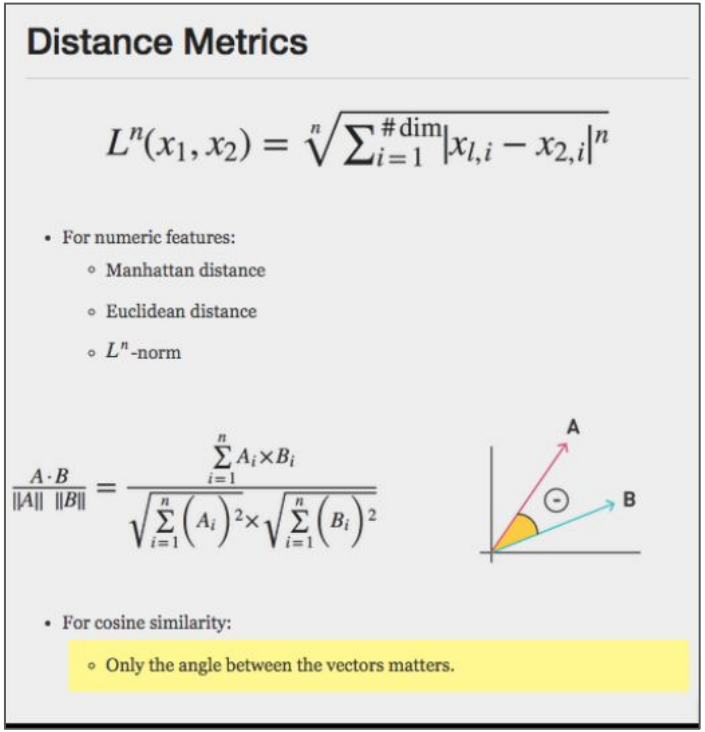
Thinking about Parameters

$$price = \alpha + \beta(bedrooms) + \gamma(bathrooms)$$

1. What are the parameters in this model?
2. What is the role of data in parametric modeling?
3. What is the role of data in non-parametric modeling?
4. What are some advantages of parametric modeling?
5. Of non-parametric modeling?
6. What are hyperparameters?

Similarity Measures

Similarity/Distance Measures



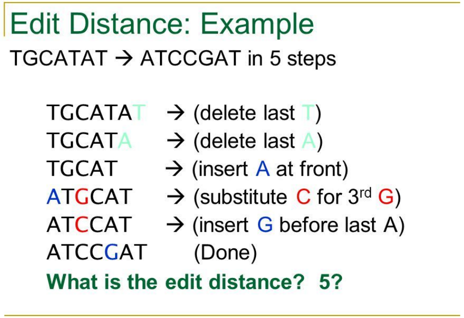
Distance Metrics

$$L^n(x_1,x_2)=\sqrt[n]{\sum_{i=1}^{\#\text{dim}}|x_{l,i}-x_{2,i}|^n}$$

- For numeric features:
    - Manhattan distance
    - Euclidean distance
    - L^n-norm

$$\frac{A\cdot B}{\|A\|\ \|B\|}=\frac{\sum\limits_{i=1}^{n}A_i\times B_i}{\sqrt{\sum\limits_{i=1}^{n}(A_i)^2}\times\sqrt{\sum\limits_{i=1}^{n}(B_i)^2}}$$

- For cosine similarity:
    - Only the angle between the vectors matters.



Edit Distance: Example

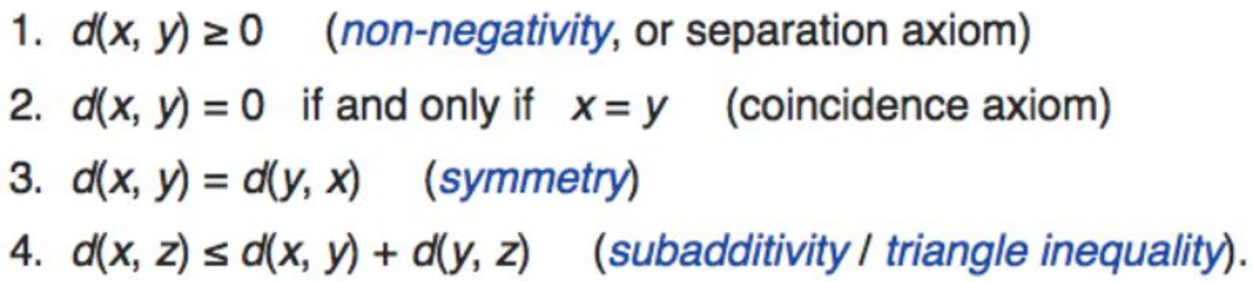
TGCATAT → ATCCGAT in 5 steps

TGCATAT → (delete last T)
TGCATA → (delete last A)
TGCAT → (insert A at front)
ATGCAT → (substitute C for 3rd G)
ATCCAT → (insert G before last A)
ATCCGAT (Done)

What is the edit distance? 5?

1. What are some similarity measures you know?
2. What is a 'distance' measure? Kernel?
3. What makes for a 'good' similarity measure?

Measures and Metrics

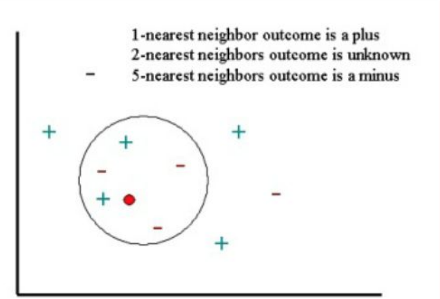


1. $d(x, y) \geq 0$ (*non-negativity*, or separation axiom)
2. $d(x, y) = 0$ if and only if $x = y$ (coincidence axiom)
3. $d(x, y) = d(y, x)$ (*symmetry*)
4. $d(x, z) \leq d(x, y) + d(y, z)$ (*subadditivity* / *triangle inequality*).

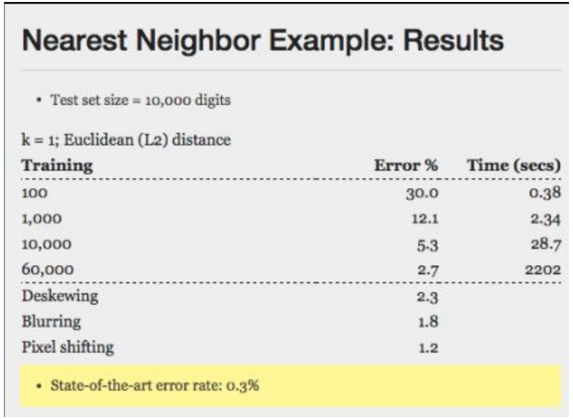
1. What is a ‘metric’? Semi-metric?
2. What is an example of a metric? Semi-metric?
3. Discuss learning a metric.

KNN

KNN Review



1-nearest neighbor outcome is a plus
2-nearest neighbors outcome is unknown
5-nearest neighbors outcome is a minus



Nearest Neighbor Example: Results

- Test set size = 10,000 digits

k = 1; Euclidean (L2) distance

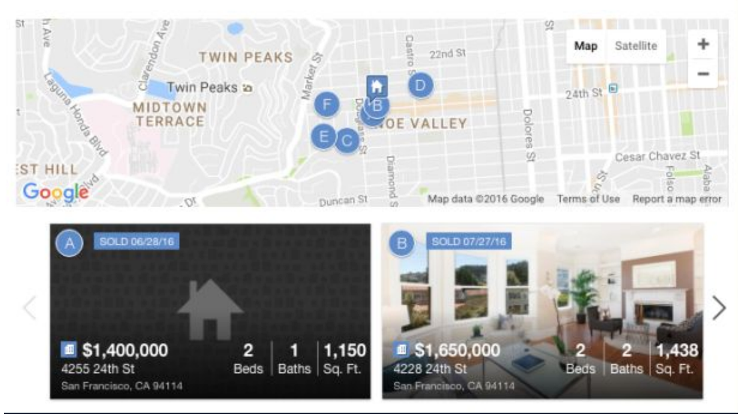
Training	Error %	Time (secs)
100	30.0	0.38
1,000	12.1	2.34
10,000	5.3	28.7
60,000	2.7	2202
Deskewing	2.3	
Blurring	1.8	
Pixel shifting	1.2	

- State-of-the-art error rate: 0.3%

1. Describe the NN Algorithm?
2. What changes with K-NN?
3. How much memory is used to store the trained model?
4. How computationally fast are predictions?
5. In general, what is the difference between regression and classification problems?
6. How can KNN be used to solve regression problems?

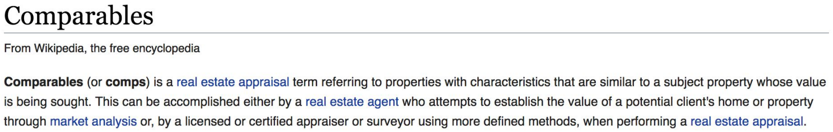
Natural Domains for KNN

1. When have you seen people naturally use NN-like algorithms?
2. From a data perspective, when do you think you might want to use NN?
3. Why do you think it's common for state-of-the-art realtime facial recognition systems to use NN?



A SOLD 06/28/16
$1,400,000
4255 24th St
San Francisco, CA 94114
2 Beds | 1 Baths | 1,150 Sq. Ft.

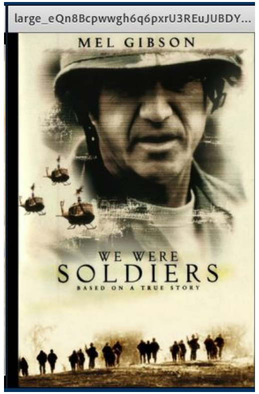
B SOLD 07/27/16
$1,650,000
4228 24th St
San Francisco, CA 94114
2 Beds | 2 Baths | 1,438 Sq. Ft.



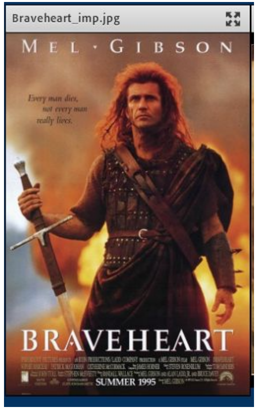
Comparables

From Wikipedia, the free encyclopedia

Comparables (or **comps**) is a real estate appraisal term referring to properties with characteristics that are similar to a subject property whose value is being sought. This can be accomplished either by a real estate agent who attempts to establish the value of a potential client's home or property through market analysis or, by a licensed or certified appraiser or surveyor using more defined methods, when performing a real estate appraisal.



=



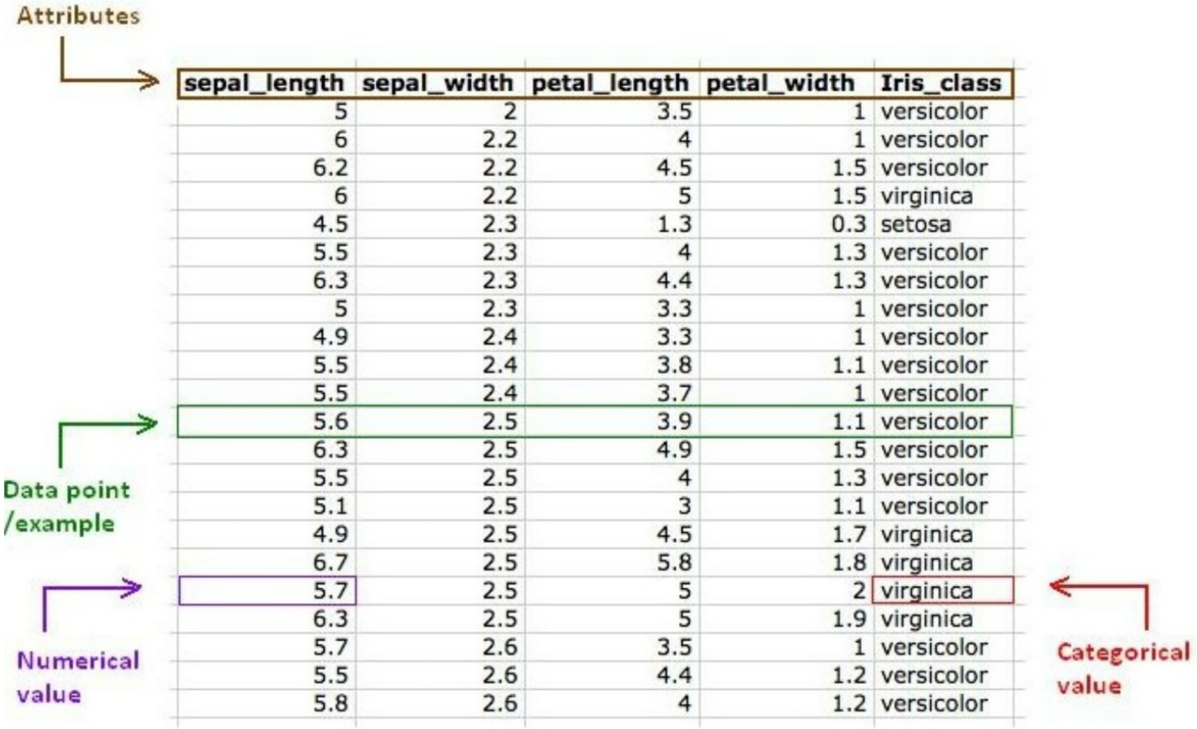
+

MARLON BRANDO
ROBERT DUVALL
MARTIN SHEEN
LAURENCE FISHBURNE
DENNIS HOPPER
HARRISON FORD
Apocalypse Now

YOLO

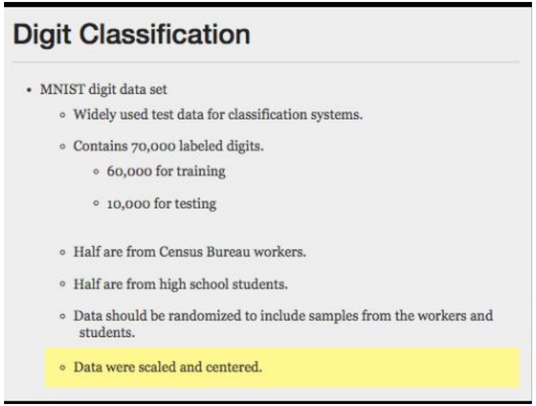
Machine Learning Workflows

Design Matrix



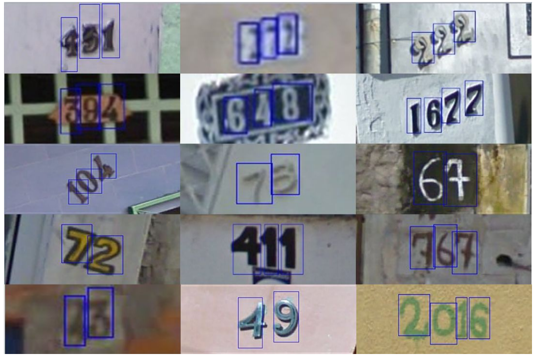
sepal_length	sepal_width	petal_length	petal_width	Iris_class
5	2	3.5	1	versicolor
6	2.2	4	1	versicolor
6.2	2.2	4.5	1.5	versicolor
6	2.2	5	1.5	virginica
4.5	2.3	1.3	0.3	setosa
5.5	2.3	4	1.3	versicolor
6.3	2.3	4.4	1.3	versicolor
5	2.3	3.3	1	versicolor
4.9	2.4	3.3	1	versicolor
5.5	2.4	3.8	1.1	versicolor
5.5	2.4	3.7	1	versicolor
5.6	2.5	3.9	1.1	versicolor
6.3	2.5	4.9	1.5	versicolor
5.5	2.5	4	1.3	versicolor
5.1	2.5	3	1.1	versicolor
4.9	2.5	4.5	1.7	virginica
6.7	2.5	5.8	1.8	virginica
5.7	2.5	5	2	virginica
6.3	2.5	5	1.9	virginica
5.7	2.6	3.5	1	versicolor
5.5	2.6	4.4	1.2	versicolor
5.8	2.6	4	1.2	versicolor

Building Datasets



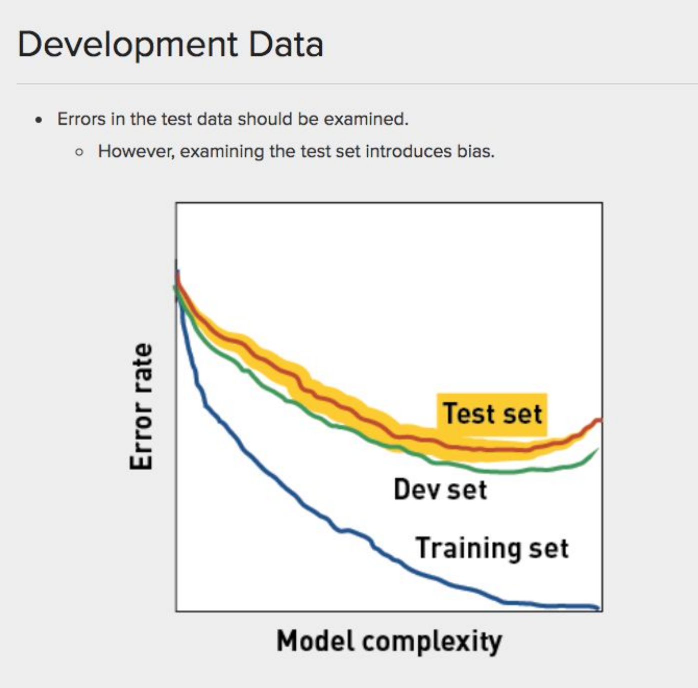
Digit Classification

- MNIST digit data set
  - Widely used test data for classification systems.
  - Contains 70,000 labeled digits.
    - 60,000 for training
    - 10,000 for testing
  - Half are from Census Bureau workers.
  - Half are from high school students.
  - Data should be randomized to include samples from the workers and students.
  - Data were scaled and centered.



1. What makes MNIST a good/bad benchmark dataset?
2. How would you create a benchmark dataset for home price prediction?
3. How would you create a benchmark dataset for speech recognition?

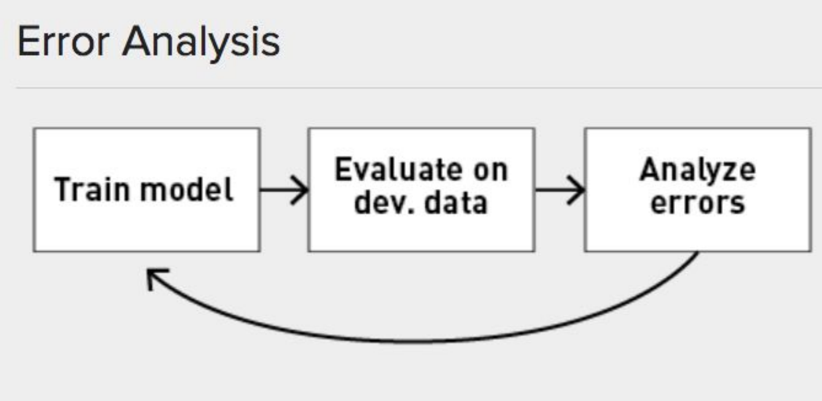
Workflow



Development Data

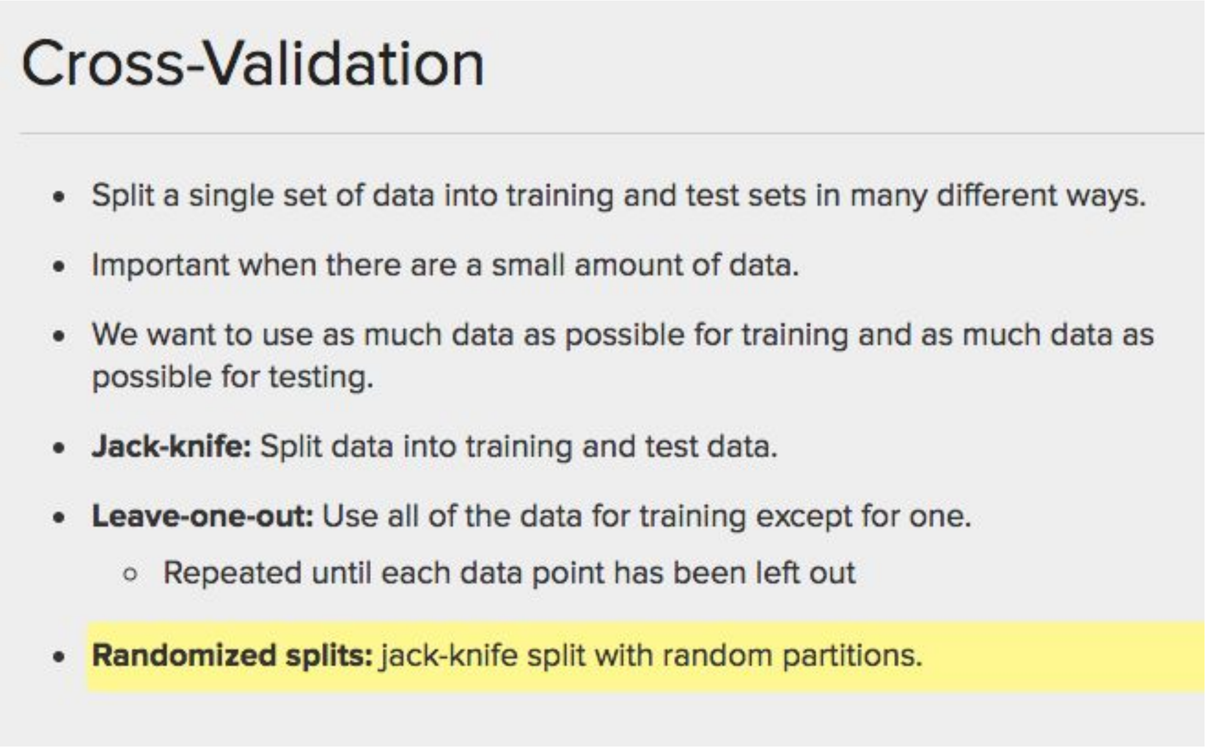
- Errors in the test data should be examined.
    - However, examining the test set introduces bias.

1. What is the role of a dev dataset?
2. What is the role of error analysis?
3. What makes for a good benchmark dataset?
4. What makes for a good baseline model?



Error Analysis

Train model → Evaluate on dev. data → Analyze errors

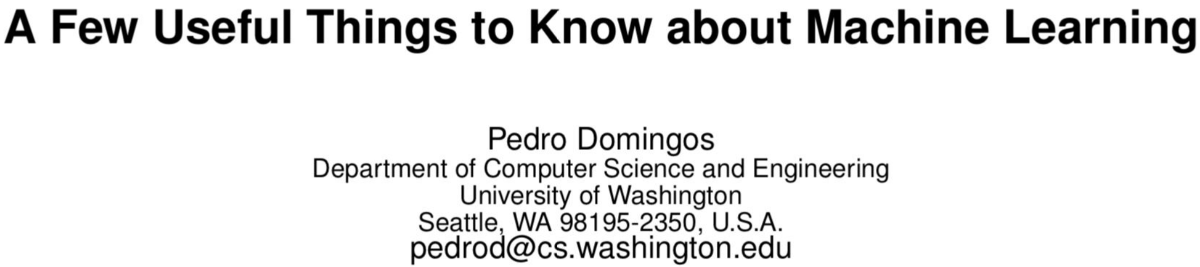


Cross-Validation

- Split a single set of data into training and test sets in many different ways.
- Important when there are a small amount of data.
- We want to use as much data as possible for training and as much data as possible for testing.
- **Jack-knife:** Split data into training and test data.
- **Leave-one-out:** Use all of the data for training except for one.
  - Repeated until each data point has been left out
- **Randomized splits:** jack-knife split with random partitions.

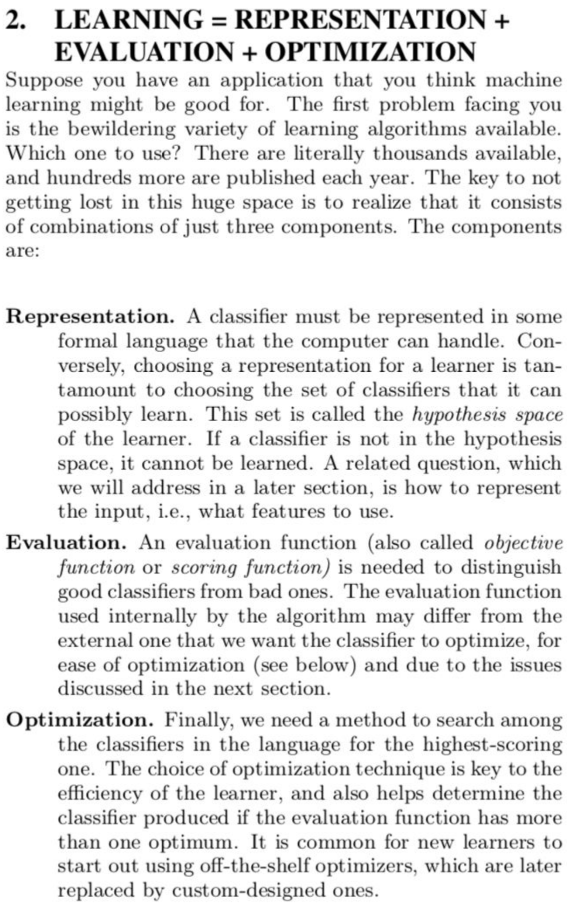
1. When might you consider using cross-validation?
2. How do you deploy a model Developed using CV?

Domingos Paper Discussion



A Few Useful Things to Know about Machine Learning

Pedro Domingos
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350, U.S.A.
pedrod@cs.washington.edu



2. LEARNING = REPRESENTATION + EVALUATION + OPTIMIZATION

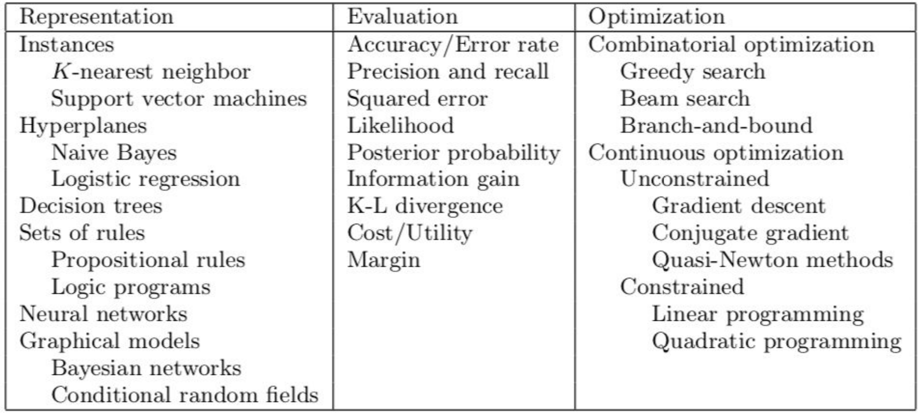
Suppose you have an application that you think machine learning might be good for. The first problem facing you is the bewildering variety of learning algorithms available. Which one to use? There are literally thousands available, and hundreds more are published each year. The key to not getting lost in this huge space is to realize that it consists of combinations of just three components. The components are:

Representation. A classifier must be represented in some formal language that the computer can handle. Conversely, choosing a representation for a learner is tantamount to choosing the set of classifiers that it can possibly learn. This set is called the *hypothesis space* of the learner. If a classifier is not in the hypothesis space, it cannot be learned. A related question, which we will address in a later section, is how to represent the input, i.e., what features to use.

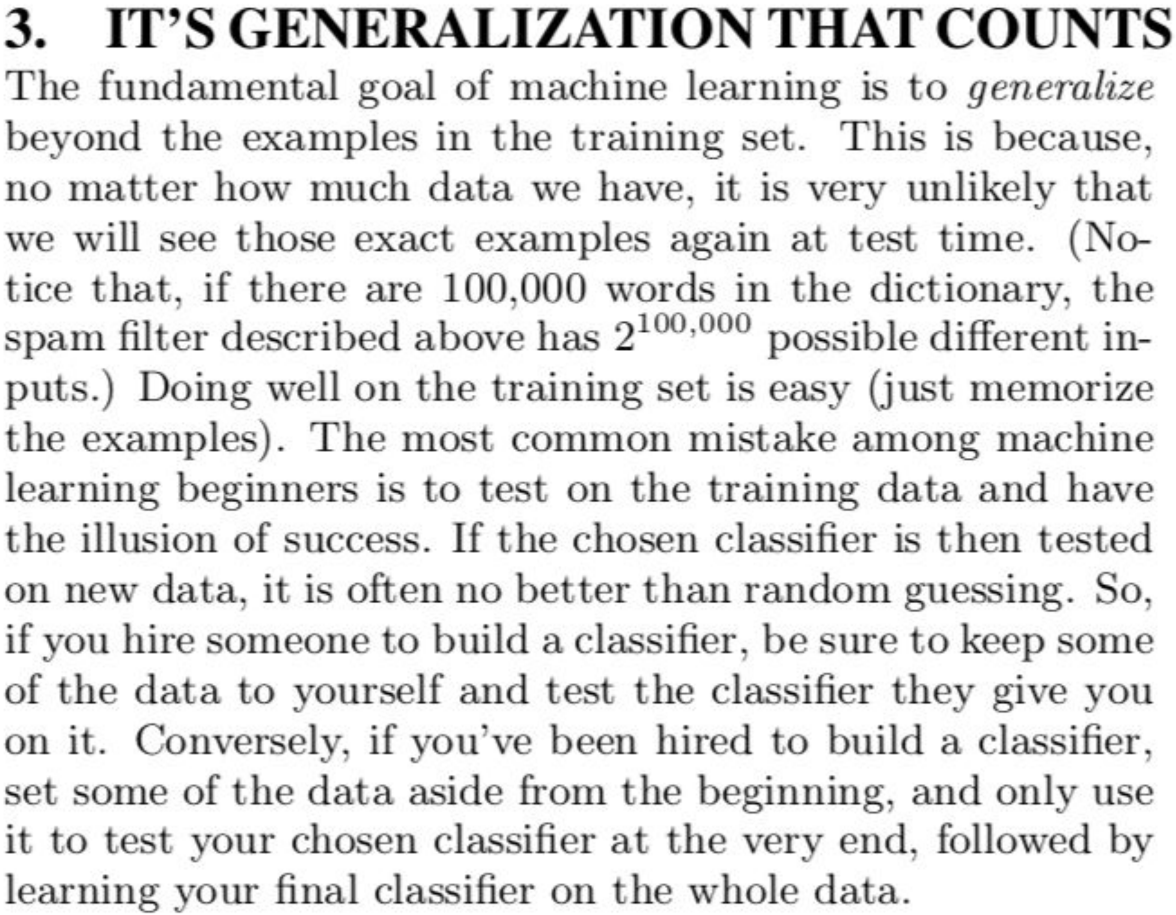
Evaluation. An evaluation function (also called *objective function* or *scoring function)* is needed to distinguish good classifiers from bad ones. The evaluation function used internally by the algorithm may differ from the external one that we want the classifier to optimize, for ease of optimization (see below) and due to the issues discussed in the next section.

Optimization. Finally, we need a method to search among the classifiers in the language for the highest-scoring one. The choice of optimization technique is key to the efficiency of the learner, and also helps determine the classifier produced if the evaluation function has more than one optimum. It is common for new learners to start out using off-the-shelf optimizers, which are later replaced by custom-designed ones.

Table 1: The three components of learning algorithms.

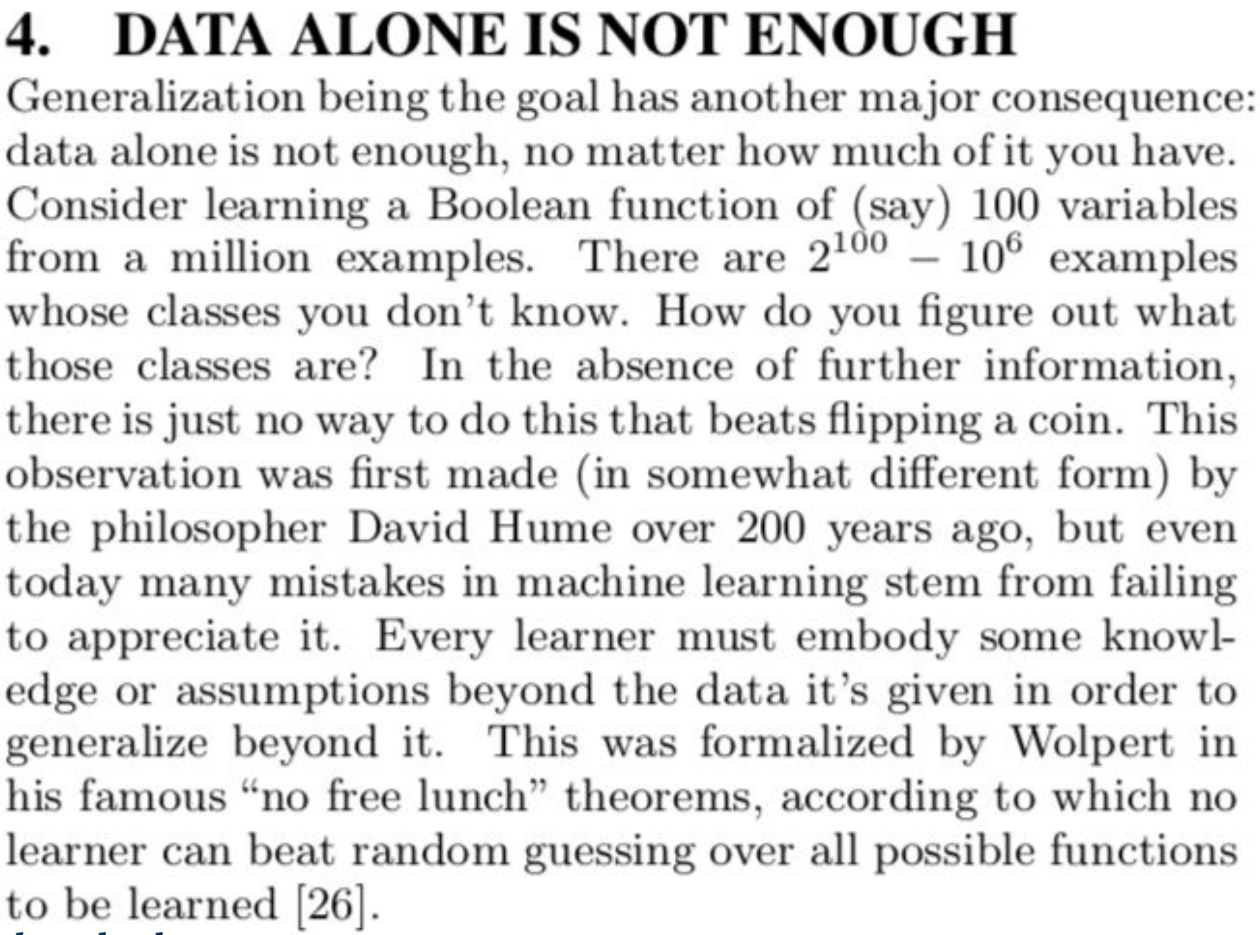


Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
K-nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search
Hyperplanes	Likelihood	Branch-and-bound
Naive Bayes	Posterior probability	Continuous optimization
Logistic regression	Information gain	Unconstrained
Decision trees	K-L divergence	Gradient descent
Sets of rules	Cost/Utility	Conjugate gradient
Propositional rules	Margin	Quasi-Newton methods
Logic programs		Constrained
Neural networks		Linear programming
Graphical models		Quadratic programming
Bayesian networks		
Conditional random fields		



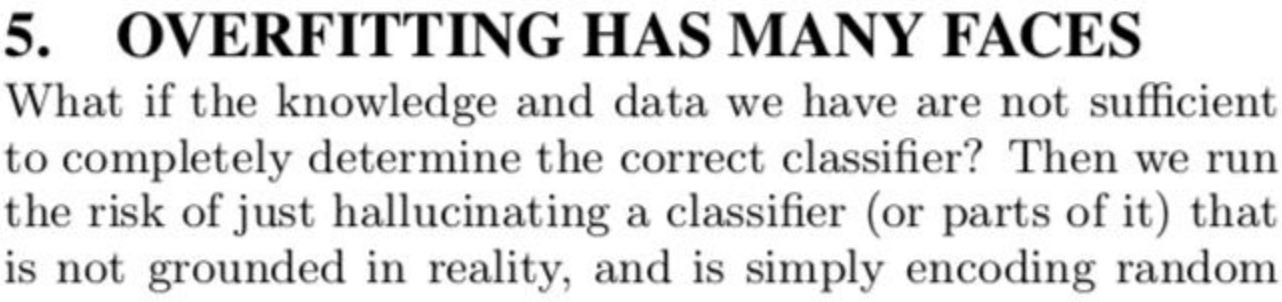
3. IT'S GENERALIZATION THAT COUNTS

The fundamental goal of machine learning is to *generalize* beyond the examples in the training set. This is because, no matter how much data we have, it is very unlikely that we will see those exact examples again at test time. (Notice that, if there are 100,000 words in the dictionary, the spam filter described above has $2^{100,000}$ possible different inputs.) Doing well on the training set is easy (just memorize the examples). The most common mistake among machine learning beginners is to test on the training data and have the illusion of success. If the chosen classifier is then tested on new data, it is often no better than random guessing. So, if you hire someone to build a classifier, be sure to keep some of the data to yourself and test the classifier they give you on it. Conversely, if you've been hired to build a classifier, set some of the data aside from the beginning, and only use it to test your chosen classifier at the very end, followed by learning your final classifier on the whole data.



4. DATA ALONE IS NOT ENOUGH

Generalization being the goal has another major consequence: data alone is not enough, no matter how much of it you have. Consider learning a Boolean function of (say) 100 variables from a million examples. There are $2^{100} - 10^6$ examples whose classes you don't know. How do you figure out what those classes are? In the absence of further information, there is just no way to do this that beats flipping a coin. This observation was first made (in somewhat different form) by the philosopher David Hume over 200 years ago, but even today many mistakes in machine learning stem from failing to appreciate it. Every learner must embody some knowledge or assumptions beyond the data it's given in order to generalize beyond it. This was formalized by Wolpert in his famous "no free lunch" theorems, according to which no learner can beat random guessing over all possible functions to be learned [26].



5. OVERFITTING HAS MANY FACES

What if the knowledge and data we have are not sufficient to completely determine the correct classifier? Then we run the risk of just hallucinating a classifier (or parts of it) that is not grounded in reality, and is simply encoding random

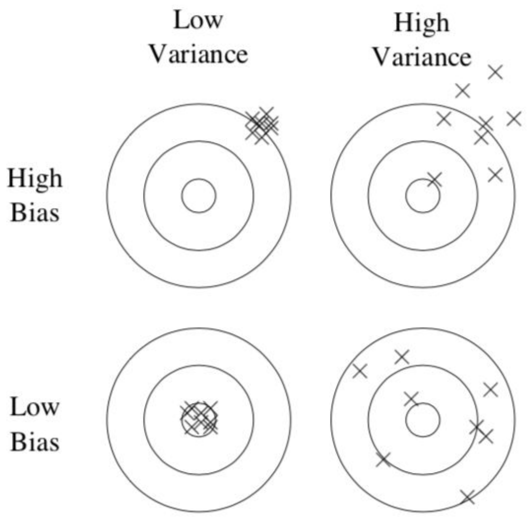
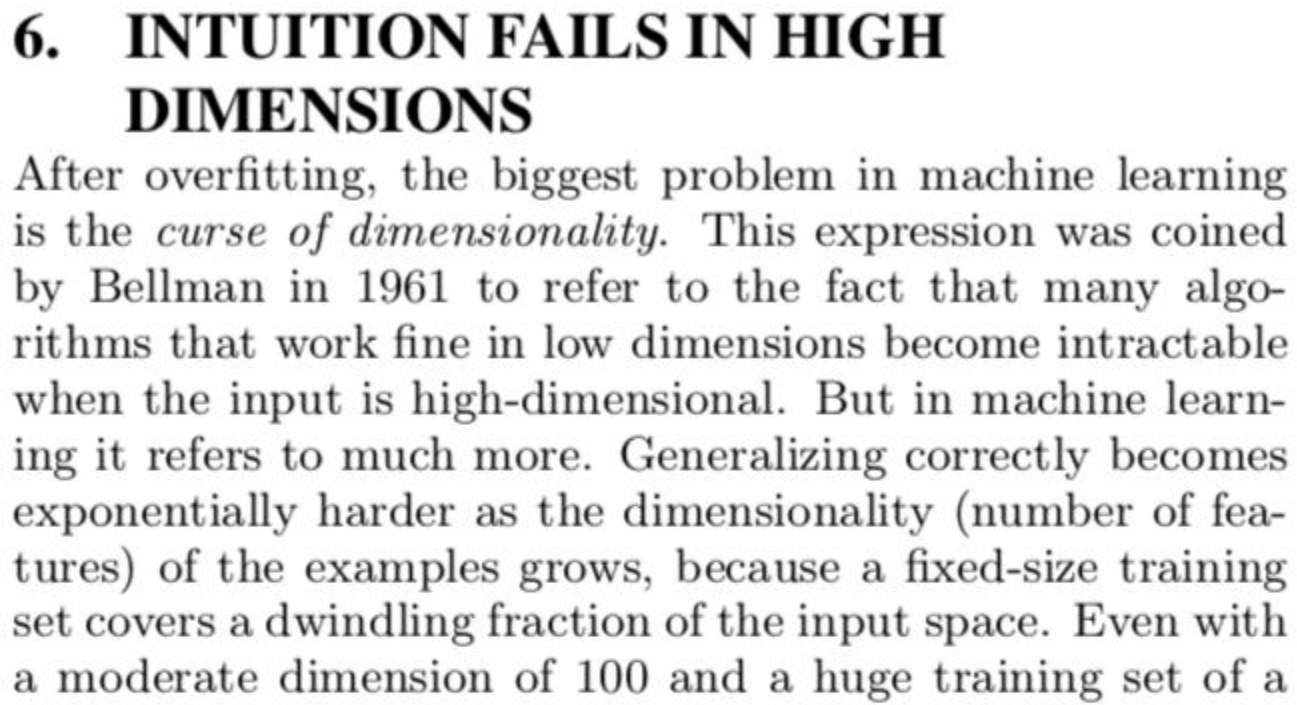
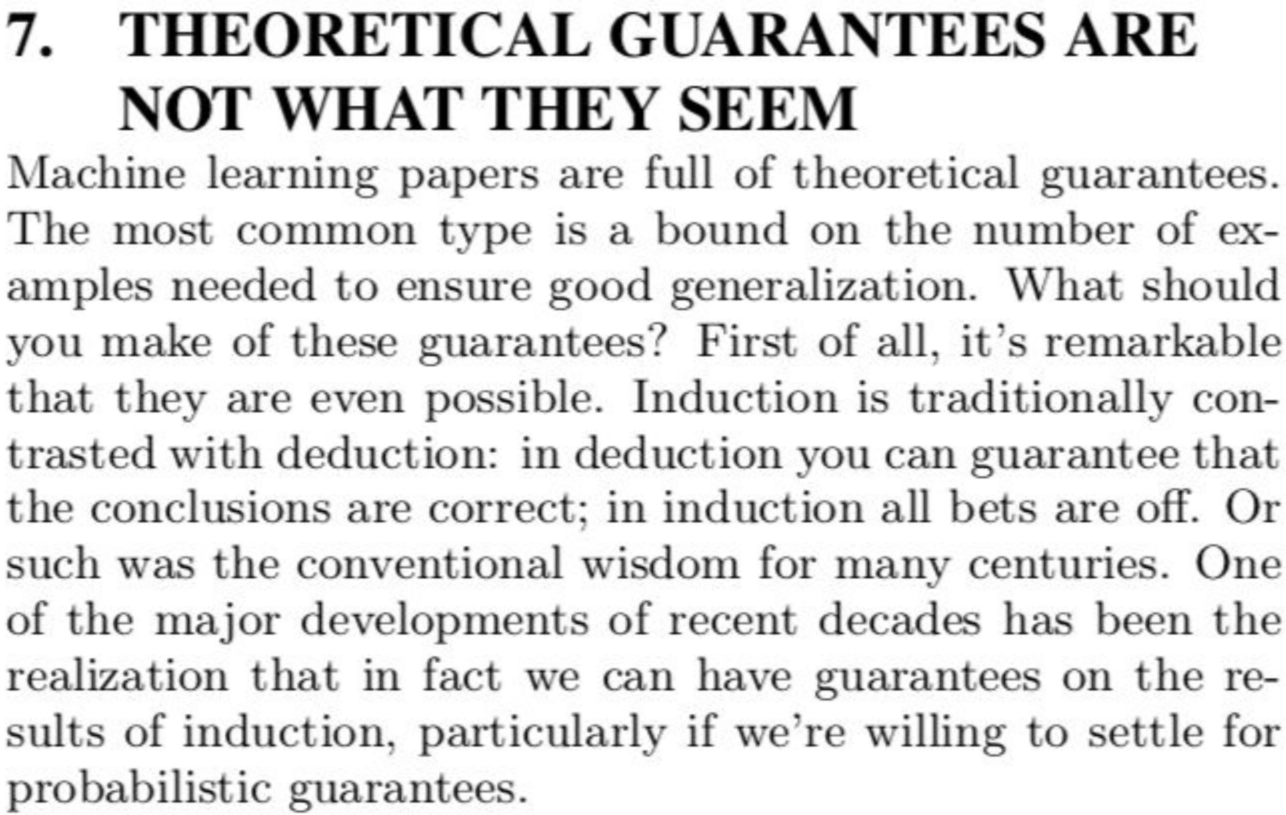


Figure 1: Bias and variance in dart-throwing.



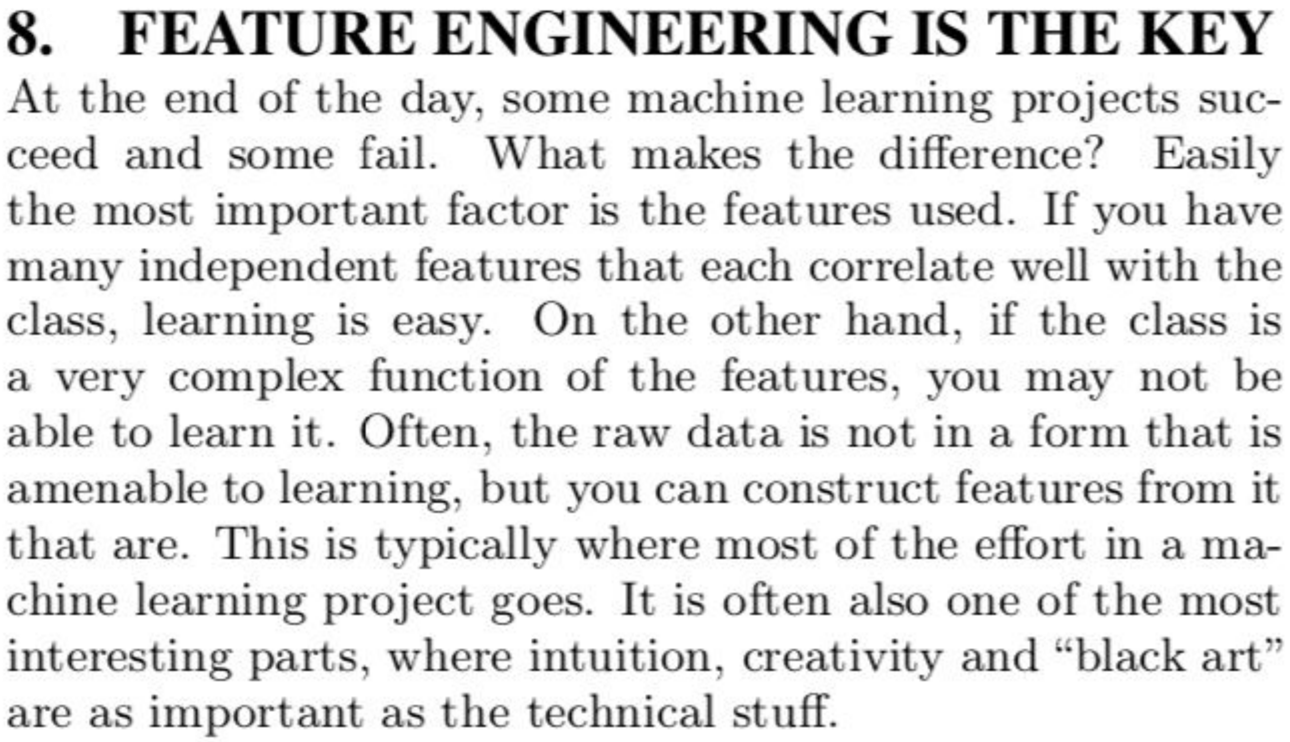
6. INTUITION FAILS IN HIGH DIMENSIONS

After overfitting, the biggest problem in machine learning is the *curse of dimensionality*. This expression was coined by Bellman in 1961 to refer to the fact that many algorithms that work fine in low dimensions become intractable when the input is high-dimensional. But in machine learning it refers to much more. Generalizing correctly becomes exponentially harder as the dimensionality (number of features) of the examples grows, because a fixed-size training set covers a dwindling fraction of the input space. Even with a moderate dimension of 100 and a huge training set of a



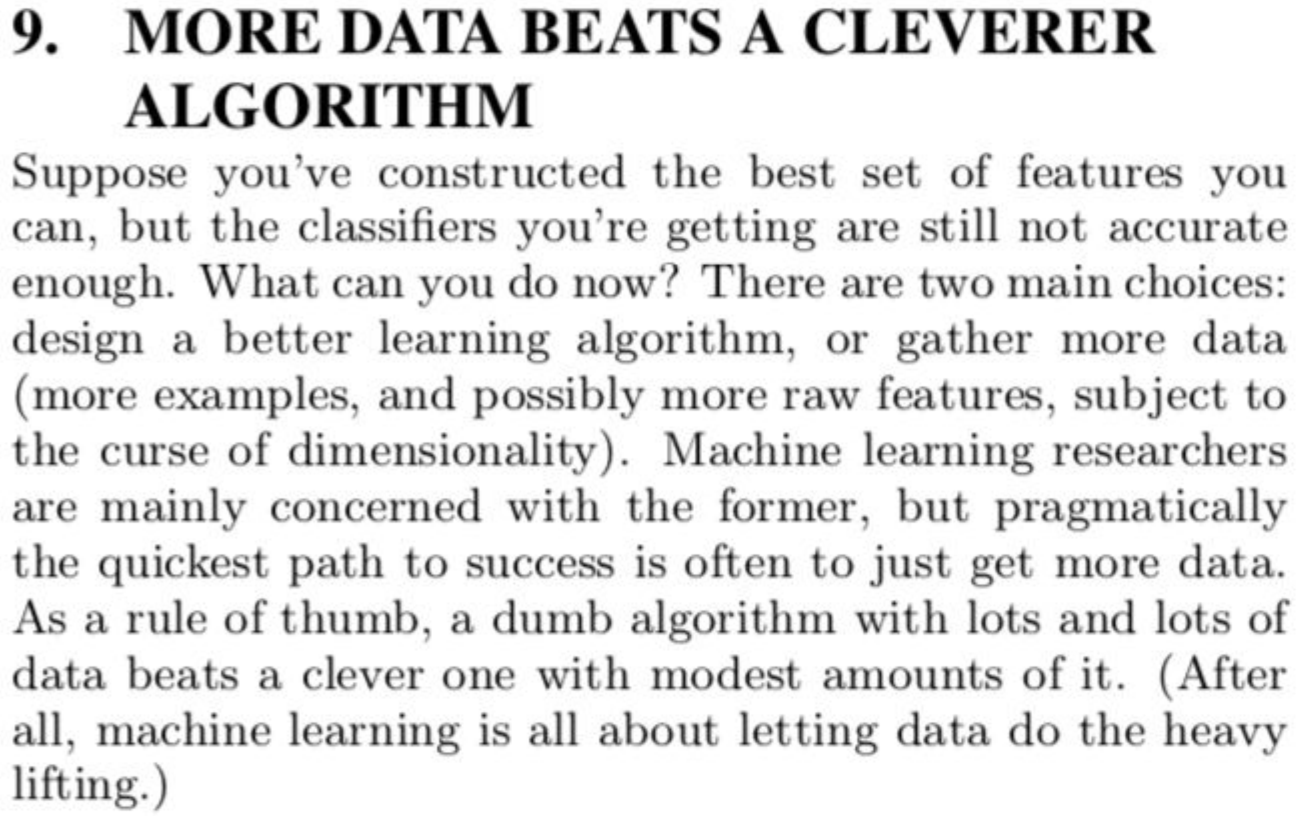
7. THEORETICAL GUARANTEES ARE NOT WHAT THEY SEEM

Machine learning papers are full of theoretical guarantees. The most common type is a bound on the number of examples needed to ensure good generalization. What should you make of these guarantees? First of all, it's remarkable that they are even possible. Induction is traditionally contrasted with deduction: in deduction you can guarantee that the conclusions are correct; in induction all bets are off. Or such was the conventional wisdom for many centuries. One of the major developments of recent decades has been the realization that in fact we can have guarantees on the results of induction, particularly if we're willing to settle for probabilistic guarantees.



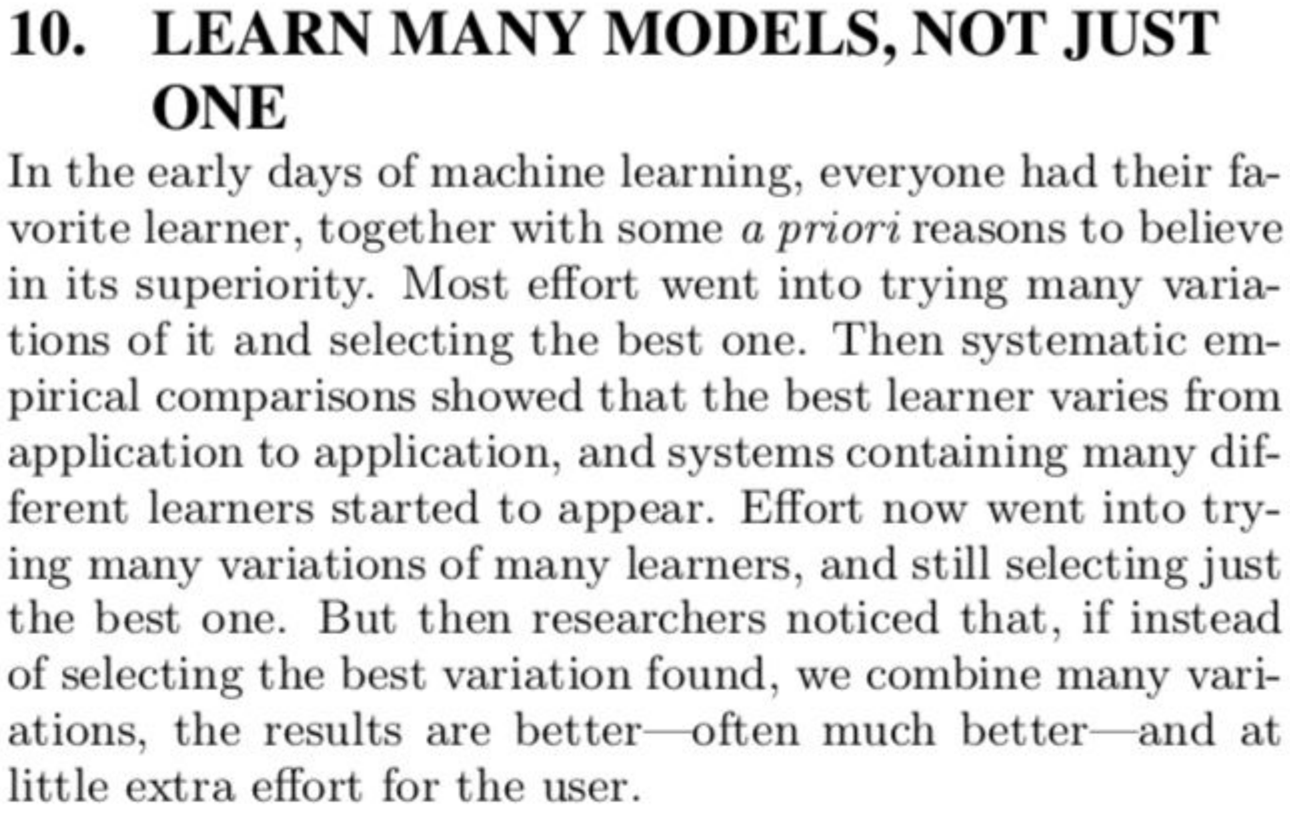
8. FEATURE ENGINEERING IS THE KEY

At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used. If you have many independent features that each correlate well with the class, learning is easy. On the other hand, if the class is a very complex function of the features, you may not be able to learn it. Often, the raw data is not in a form that is amenable to learning, but you can construct features from it that are. This is typically where most of the effort in a machine learning project goes. It is often also one of the most interesting parts, where intuition, creativity and "black art" are as important as the technical stuff.



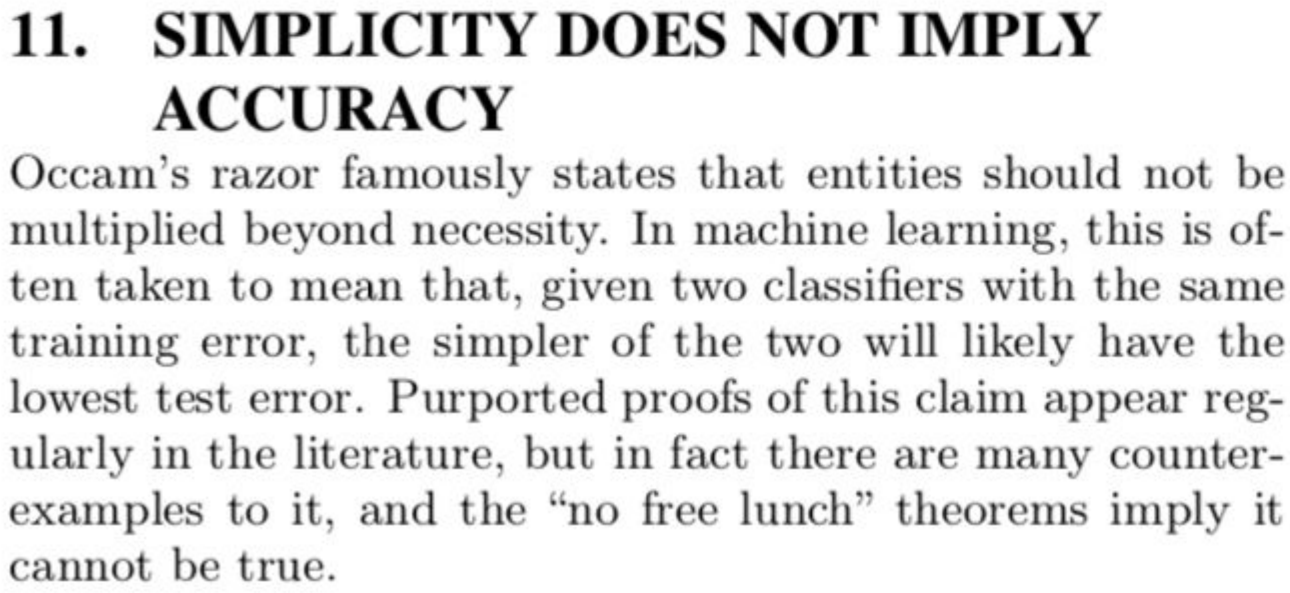
9. MORE DATA BEATS A CLEVERER ALGORITHM

Suppose you've constructed the best set of features you can, but the classifiers you're getting are still not accurate enough. What can you do now? There are two main choices: design a better learning algorithm, or gather more data (more examples, and possibly more raw features, subject to the curse of dimensionality). Machine learning researchers are mainly concerned with the former, but pragmatically the quickest path to success is often to just get more data. As a rule of thumb, a dumb algorithm with lots and lots of data beats a clever one with modest amounts of it. (After all, machine learning is all about letting data do the heavy lifting.)



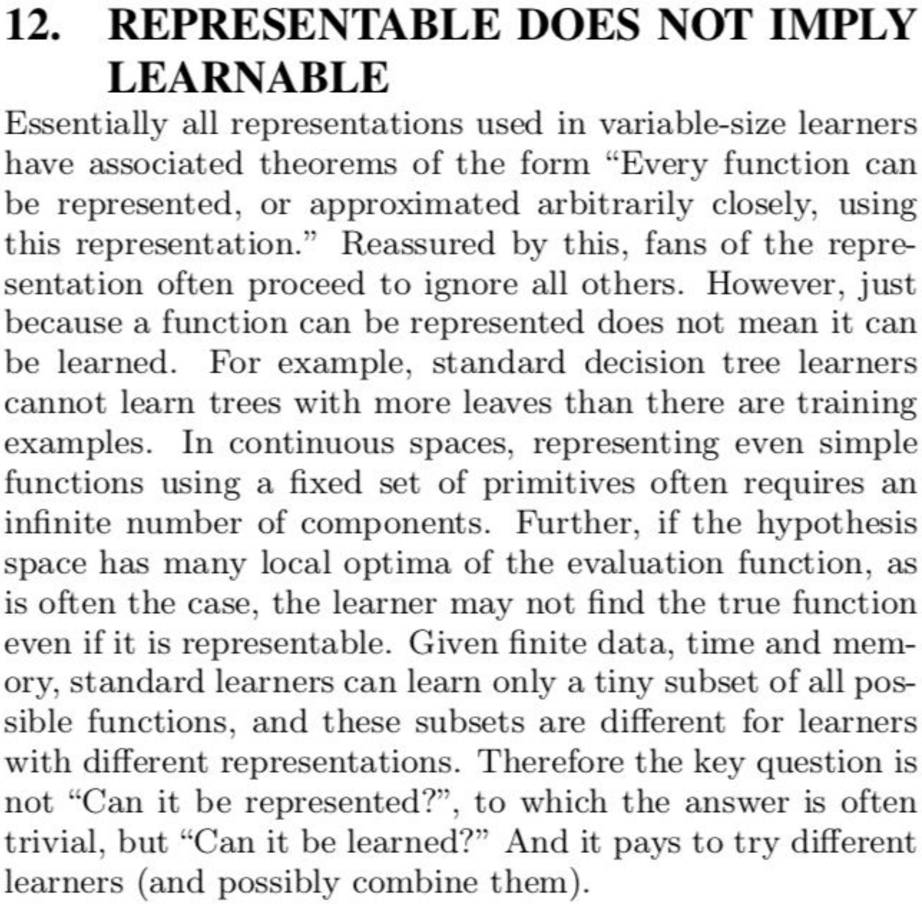
10. LEARN MANY MODELS, NOT JUST ONE

In the early days of machine learning, everyone had their favorite learner, together with some *a priori* reasons to believe in its superiority. Most effort went into trying many variations of it and selecting the best one. Then systematic empirical comparisons showed that the best learner varies from application to application, and systems containing many different learners started to appear. Effort now went into trying many variations of many learners, and still selecting just the best one. But then researchers noticed that, if instead of selecting the best variation found, we combine many variations, the results are better—often much better—and at little extra effort for the user.



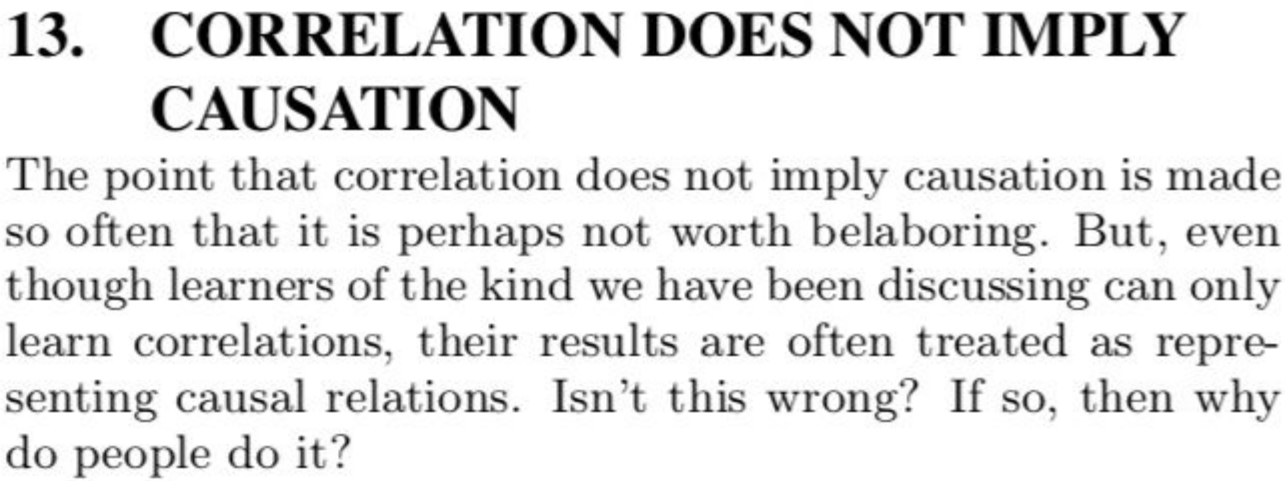
11. SIMPLICITY DOES NOT IMPLY ACCURACY

Occam's razor famously states that entities should not be multiplied beyond necessity. In machine learning, this is often taken to mean that, given two classifiers with the same training error, the simpler of the two will likely have the lowest test error. Purported proofs of this claim appear regularly in the literature, but in fact there are many counterexamples to it, and the "no free lunch" theorems imply it cannot be true.



12. REPRESENTABLE DOES NOT IMPLY LEARNABLE

Essentially all representations used in variable-size learners have associated theorems of the form “Every function can be represented, or approximated arbitrarily closely, using this representation.” Reassured by this, fans of the representation often proceed to ignore all others. However, just because a function can be represented does not mean it can be learned. For example, standard decision tree learners cannot learn trees with more leaves than there are training examples. In continuous spaces, representing even simple functions using a fixed set of primitives often requires an infinite number of components. Further, if the hypothesis space has many local optima of the evaluation function, as is often the case, the learner may not find the true function even if it is representable. Given finite data, time and memory, standard learners can learn only a tiny subset of all possible functions, and these subsets are different for learners with different representations. Therefore the key question is not “Can it be represented?”, to which the answer is often trivial, but “Can it be learned?” And it pays to try different learners (and possibly combine them).



13. CORRELATION DOES NOT IMPLY CAUSATION

The point that correlation does not imply causation is made so often that it is perhaps not worth belaboring. But, even though learners of the kind we have been discussing can only learn correlations, their results are often treated as representing causal relations. Isn't this wrong? If so, then why do people do it?

Final Thoughts?