Stat 205: Introduction to Nonparametric Statistics

Lecture 08: Predictive Modeling Overview

Instructor David Donoho; TA: Yu Wang

What is Nonparametric?

Nonparametric = nonParametric.

- Not $N(\mu, \sigma^2)$
- Not $Y = \alpha + \beta X + Z$

Two meanings, two very different use cases.

Inference techniques offering valid p-values and confidence statements under minimal assumptions: sign test, median test, Wilcoxon's tests, and the Kruskal-Wallis and Friedman tests, tests of independence.

Predictive modeling techniques valid quite generally, such as kernel and spline smoothing, nearest neighbor, and even deep neural networks.

What is Nonparametric?

Nonparametric = nonParametric.

- Not $N(\mu, \sigma^2)$
- Not $Y = \alpha + \beta X + Z$

Two meanings, two very different use cases.

Inference techniques offering valid p-values and confidence statements under minimal assumptions: sign test, median test, Wilcoxon's tests, and the Kruskal-Wallis and Friedman tests, tests of independence. 2016 Stat 205

Predictive modeling techniques valid quite generally, such as kernel and spline smoothing, nearest neighbor, and even deep neural networks. 2021 Stat 205

Use Cases for Predictive Modeling

- Density Estimation

$$X_1, \ldots, X_n$$

$$P(X \in dx) \approx f(x)dx.$$

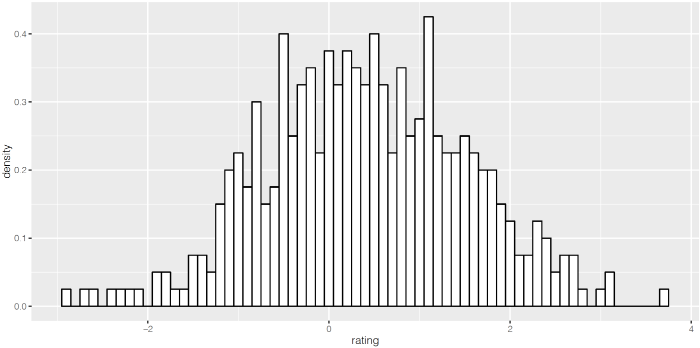
  - Typical Behaviors?
  - Underlying Clusters?
  - Extreme Behaviors?
- Nonparametric Regression

$$(X_1, Y_1), \ldots, (X_n, Y_n)$$

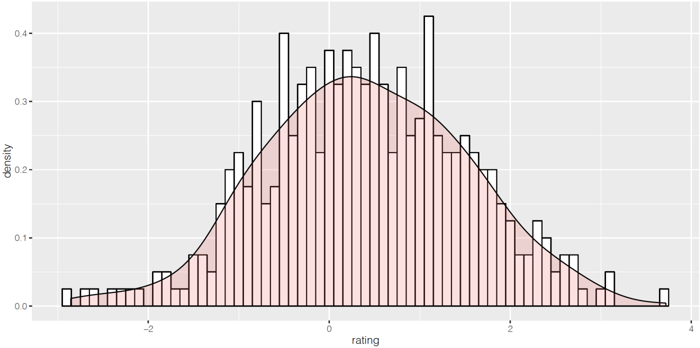
$$Y_i \approx f(X_i)$$

  - Predict
  - Understand
  - Control
- **AKA Machine Learning**
  - Density Estimation $\Leftrightarrow$ Unsupervised ML
  - Nonparametric Regression $\Leftrightarrow$ Supervised ML

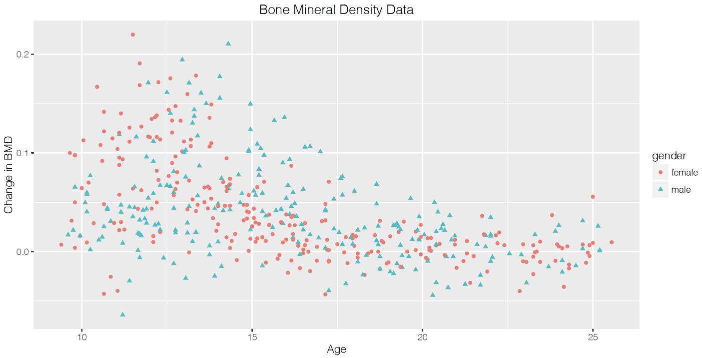
Density Estimation



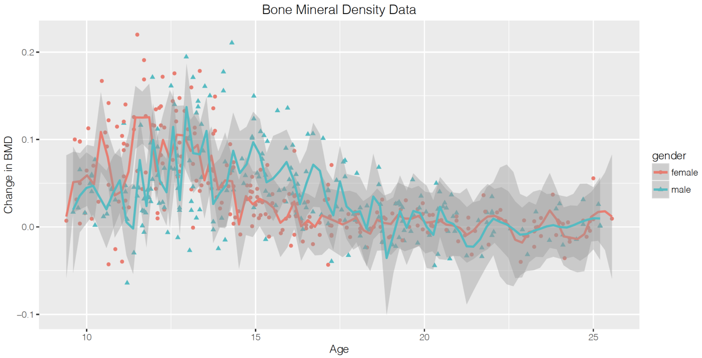
Density Estimation



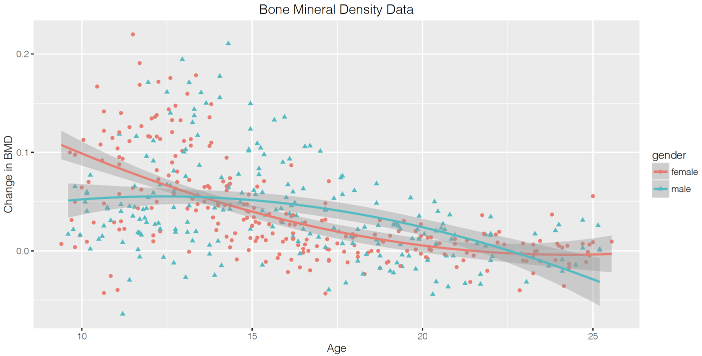
Nonlinear Regression



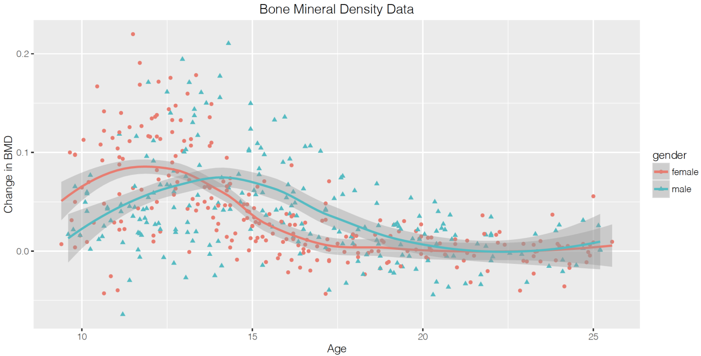
Nonlinear Regression



Nonlinear Regression



Nonlinear Regression



Rigid [Parametric] Predictive Modeling

Examples:

- Simple Linear Regression: $Y = \alpha + \beta X + Z$, $\alpha, \beta \in \mathbf{R}$.
- Multiple Linear Regression: $Y = X\beta + Z$, $\beta \in \mathbf{R}^p$.
- Principal Components: $Y = \gamma u v' + Z$, $v \in \mathbf{R}^p$, $\gamma \in \mathbf{R}$, $u \in \mathbf{R}^n$.

Advantages:

- Few Parameters
- Efficient Algorithms
- Efficient Storage
- Efficient Communication

Disadvantages

- Might be inaccurate

Flexible [nonParametric] Predictive Modeling

Examples:

- Kernel Methods $\mu(x) = \frac{\sum_i y_i K(x-x_i)}{\sum_i K(x-x_i)}$
- Spline Methods $\mu(x) = \sum_j c_j B_j(x)$
- Neural Nets $\mu(x) = \sum_j d_j \rho_j(u_j'(x - x_j))$
-

Advantages:

- Stay 'Close to the data'
- Stay 'General'

Disadvantages

- Computationally demanding

The Central Tradeoff: Flexibility vs. Parsimony, 1

Property	Approach Name	Example	#Parameters
Parsimony	Parametric Model	$Y \approx \alpha + \beta X$	2
Flexibility	nonparametric Model	$\mu(x) = \frac{\sum_i y_i K(x - x_i)}{\sum_i K(x - x_i)}$	n

Hence 'nonparametric' may actually mean 'very many parameters'

The Central Tradeoff: Flexibility vs. Parsimony, 2

Property	Approach Name	Problems	Meaning
Parsimony	Parametric Model	Bias	Systematically W
Flexibility	nonparametric Model	Variance	Largely Nois

The Central Tradeoff: Flexibility vs. Parsimony, 3

Property	Approach Name	Problem	Meaning
Parsimony	Parametric Model	Bias	Systematically Wrong
Flexibility	nonparametric Model	Variance	Largely Noise
		Inconsistency	Too Wiggly in X
		Stability	Tiny Changes in database Huge Effect

The Central Tradeoff: Flexibility vs. Parsimony, 4

- Every 'flexible fitting' method has a *tuning parameter*
  - TP controls 'Parsimony-Flexibility' Tradeoff
  - TP controls bias
    More flexibility, less bias
  - TP controls variance
    More flexibility, more variance
- Consequence *Bias-variance tradeoff*

The Central Tradeoff: Flexibility vs. Parsimony, 5

Setting	Tuning Parameter Name
Polynomial Fitting	Degree
Local Averaging	# Bins
Kernel Smoothing	Bandwidth
Spline	# Knots
Smoothing Spline	Penalty Coefficient
Orthogonal Series	# Terms
Wavelet Smoothing	Threshold
Neural Net	Width Depth Connectivity

The Central Tradeoff: Flexibility vs. Parsimony, 6

With more data:

- Variance at any given flexibility goes *down*
- Enable more flexible while still decreasing variance
- More flexibility – less bias
- Less bias *and* less variance

More data leads to better fitting models
More data changes parsimony/flexibility tradeoff
As data accumulate, always tend towards flexibility

The Bitter Lesson

The Bitter Lesson

Rich Sutton

March 13, 2019

The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on one is time not spent on the other. There are psychological commitments to investment in one approach or the other. And the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation. There were many examples of AI researchers' belated learning of this bitter lesson, and it is instructive to review some of the most prominent.

In computer chess, the methods that defeated the world champion, Kasparov, in 1997, were based on massive, deep search. At the time, this was looked upon with dismay by the majority of computer-chess researchers who had pursued methods that leveraged human understanding of the special structure of chess. When a simpler, search-based approach with special hardware and software proved vastly more effective, these human-knowledge-based chess researchers were not good losers. They said that ``brute force" search may have won this time, but it was not a general strategy, and anyway it was not how people played chess. These researchers wanted methods based on human input to win and were disappointed when they did not.

A similar pattern of research progress was seen in computer Go, only delayed by a further 20 years. Enormous initial efforts went into avoiding search by taking advantage of human knowledge, or of the special features of the game, but all those efforts proved irrelevant, or worse, once search was applied effectively at scale. Also important was the use of learning by self play to learn a value function (as it was in many other games and even in chess, although learning did not play a big role in the 1997 program that first beat a world champion). Learning by self play, and learning in general, is like search in that it enables massive computation to be brought to bear. Search and learning are the two most important classes of techniques for utilizing massive amounts of computation in AI research. In computer Go, as in computer chess, researchers' initial effort was directed towards utilizing human understanding (so that less search was needed) and only much later was much greater success had by embracing search and learning.

In speech recognition, there was an early competition, sponsored by DARPA, in the 1970s. Entrants included a host of special methods that took advantage of human knowledge---knowledge of words, of phonemes, of the human vocal tract, etc. On the other side were newer methods that were more statistical in nature and did much more computation, based on hidden Markov models (HMMs). Again, the statistical methods won out over the human-knowledge-based methods. This led to a major change in all of natural language processing, gradually over decades, where statistics and computation came to dominate the field. The recent rise of deep learning in speech recognition is the most recent step in this consistent direction. Deep learning methods rely even less on human knowledge, and use even more computation, together with learning on huge training sets, to produce dramatically better speech recognition systems. As in the games, researchers always tried to make systems that worked the way the researchers thought their own minds worked---they tried to put that knowledge in their systems---but it proved ultimately counterproductive, and a colossal waste of researcher's time, when, through Moore's law, massive computation became available and a means was found to put it to good use.

In computer vision, there has been a similar pattern. Early methods conceived of vision as searching for edges, or generalized cylinders, or in terms of SIFT features. But today all this is discarded. Modern deep-learning neural networks use only the notions of convolution and certain kinds of invariances, and perform much better.

This is a big lesson. As a field, we still have not thoroughly learned it, as we are continuing to make the same kind of mistakes. To see this, and to effectively resist it, we have to understand the appeal of these mistakes. We have to learn the bitter lesson that building in how we think we think does not work in the long run. The bitter lesson is based on the historical observations that 1) AI researchers have often tried to build knowledge into their agents, 2) this always helps in the short term, and is personally satisfying to the researcher, but 3) in the long run it plateaus and even inhibits further progress, and 4) breakthrough progress eventually arrives by an opposing approach based on scaling computation by search and learning. The eventual success is tinged with bitterness, and often incompletely digested, because it is success over a favored, human-centric approach.

One thing that should be learned from the bitter lesson is the great power of general purpose methods, of methods that continue to scale with increased computation even as the available computation becomes very great. The two methods that seem to scale arbitrarily in this way are *search* and *learning*.

The second general point to be learned from the bitter lesson is that the actual contents of minds are tremendously, irredeemably complex; we should stop trying to find simple ways to think about the contents of minds, such as simple ways to think about space, objects, multiple agents, or symmetries. All these are part of the arbitrary, intrinsically-complex, outside world. They are not what should be built in, as their complexity is endless; instead we should build in only the meta-methods that can find and capture this arbitrary complexity. Essential to these methods is that they can find good approximations, but the search for them should be by our methods, not by us. We want AI agents that

The Bitter Lesson, 2

The Bitter Lesson

Rich Sutton

March 13, 2019

The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on one is time not spent on the other. There are psychological commitments to investment in one approach or the other. And the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation. There were many examples of AI researchers' belated learning of this bitter lesson, and it is instructive to review some of the most prominent.

In computer chess, the methods that defeated the world champion, Kasparov, in 1997, were based on massive, deep search. At the time, this was looked upon with dismay by the majority of computer-chess researchers who had pursued methods that leveraged human understanding of the special structure of chess. When a simpler, search-based approach with special hardware and software proved vastly more effective, these human-knowledge-based chess researchers were not good losers. They said that ``brute force" search may have won this time, but it was not a general strategy, and anyway it was not how people played chess. These researchers wanted methods based on human input to win and were disappointed when they did not.

Rich Sutton

The Bitter Lesson, 3

A similar pattern of research progress was seen in computer Go, only delayed by a further 20 years. Enormous initial efforts went into avoiding search by taking advantage of human knowledge, or of the special features of the game, but all those efforts proved irrelevant, or worse, once search was applied effectively at scale. Also important was the use of learning by self play to learn a value function (as it was in many other games and even in chess, although learning did not play a big role in the 1997 program that first beat a world champion). Learning by self play, and learning in general, is like search in that it enables massive computation to be brought to bear. Search and learning are the two most important classes of techniques for utilizing massive amounts of computation in AI research. In computer Go, as in computer chess, researchers' initial effort was directed towards utilizing human understanding (so that less search was needed) and only much later was much greater success had by embracing search and learning.

In speech recognition, there was an early competition, sponsored by DARPA, in the 1970s. Entrants included a host of special methods that took advantage of human knowledge---knowledge of words, of phonemes, of the human vocal tract, etc. On the other side were newer methods that were more statistical in nature and did much more computation, based on hidden Markov models (HMMs). Again, the statistical methods won out over the human-knowledge-based methods. This led to a major change in all of natural language processing, gradually over decades, where statistics and computation came to dominate the field. The recent rise of deep learning in speech recognition is the most recent step in this consistent direction. Deep learning methods rely even less on human knowledge, and use even more computation, together with learning on huge training sets, to produce dramatically better speech recognition systems. As in the games, researchers always tried to make systems that worked the way the researchers thought their own minds worked---they tried to put that knowledge in their systems---but it proved ultimately counterproductive, and a colossal waste of researcher's time, when, through Moore's law, massive computation became available and a means was found to put it to good use.

In computer vision, there has been a similar pattern. Early methods conceived of vision as searching for edges, or generalized cylinders, or in terms of SIFT features. But today all this is discarded. Modern deep-learning neural networks use only the notions of convolution and certain kinds of invariances, and perform much better.

Rich Sutton

The Bitter Lesson, 4

This is a big lesson. As a field, we still have not thoroughly learned it, as we are continuing to make the same kind of mistakes. To see this, and to effectively resist it, we have to understand the appeal of these mistakes. We have to learn the bitter lesson that building in how we think we think does not work in the long run. The bitter lesson is based on the historical observations that 1) AI researchers have often tried to build knowledge into their agents, 2) this always helps in the short term, and is personally satisfying to the researcher, but 3) in the long run it plateaus and even inhibits further progress, and 4) breakthrough progress eventually arrives by an opposing approach based on scaling computation by search and learning. The eventual success is tinged with bitterness, and often incompletely digested, because it is success over a favored, human-centric approach.

One thing that should be learned from the bitter lesson is the great power of general purpose methods, of methods that continue to scale with increased computation even as the available computation becomes very great. The two methods that seem to scale arbitrarily in this way are *search* and *learning*.

The second general point to be learned from the bitter lesson is that the actual contents of minds are tremendously, irredeemably complex; we should stop trying to find simple ways to think about the contents of minds, such as simple ways to think about space, objects, multiple agents, or symmetries. All these are part of the arbitrary, intrinsically-complex, outside world. They are not what should be built in, as their complexity is endless; instead we should build in only the meta-methods that can find and capture this arbitrary complexity. Essential to these methods is that they can find good approximations, but the search for them should be by our methods, not by us. We want AI agents that can discover like we can, not which contain what we have discovered. Building in our discoveries only makes it harder to see how the discovering process can be done.

Rich Sutton

The Bitter Lesson, Restated

- We ‘build in’ some ‘simplifying model’
  - It reflects some data/phenomenology we’ve seen
  - It reflects some persuasive heuristics
  - It beguiles us and others
- It helps *at first*:
  - We save compute, save data, because simplicity
  - We understand it, because we made it
  - We love it, because simplicity is beauty
- Ultimately, the once-helpful model holds us back:
  - New generations don’t need it, outperform.
  - We fell in love with it, can’t let go
  - We live ever after in bitterness.