Supervised Machine Learning I: Prediction

Paul Goldsmith-Pinkham

April 27, 2021

Machine Learning

- The next few classes we will be studying *Machine Learning*
  - This is a vague phrase that can mean many things.
  - In fact, we have already studied versions of machine learning, ranging from OLS to penalized linear models like Lasso!

- First, we'll discuss what we usually mean when discussing "ML"

- Then, today we will focus on methods related to prediction, or forecasting
  - Subsequent classes will study how to use these methods to do heterogeneity analysis and categorization of unsupervised data

Machine learning and prediction

- What tends to delineate ML as an approach is a focus on algorithms, rather than statistical processes
  - These points are emphasized by Brieman [2001] and subseqeuntly by Athey and Imbens [2019]

- The tension argued by Brieman is a focus on "theory" vs. solving practical problems
  - E.g., are we willing to use something if we don't already know it's a consistent / asymptotically normal estimator?
  - What if it works well in many samples?

- A crucial distinction that has been highlighted between the ML vs. "traditional" stats settings is the difference between *prediction* and unbiased *estimation*
  - Mullinathan and Spiess (2017) refer to $\hat{y}$ vs $\hat{\beta}$

Machine learning and prediction

- Today, we're going to focus on *prediction*. What are applications when prediction is useful?

Machine learning and prediction

- Today, we're going to focus on *prediction*. What are applications when prediction is useful?
  1. Forecasting – macro or financial.
  2. Assessing the informativeness of new data
  3. Contrasting human decision-making with algorithms
  4. Predicting valuations for unpriced goods?
  5. Others?

- The crucial concern is that prediction *is not* causal identification
  - There is no way to use more data to solve a causal inference problem
  - That's ok though! There are circumstances where better prediction can be useful to test our models

Refresher: what have we already covered?

- Recall our lecture on *Penalized Linear Models*
- We considered linear model settings with

$$Y_i = X_i\beta + \epsilon_i,$$

where we expanded on the traditional OLS estimation objective function with a penalty term:

$$\hat{\beta} = \arg\min_{\beta} \sum_i (Y_i - X_i\beta)^2 + \lambda(||\beta||_q)^{1/q}$$

- When $q = 1$, this was LASSO; with $q = 2$, this was ridge regression
  - There are a number of useful hybrid methods that combine different versions of this
  - These models vary in their success and in their computational tractability as K gets large
  - LASSO and ridge (and elastic net) have algorithmically had much success due to their solvability

Refresher: what have we already covered?

$$\hat{\beta} = \arg\min_{\beta} \sum_{i} (Y_i - X_i\beta)^2 + \lambda(||\beta||_q)^{1/q}$$

- This is already “Machine Learning”
- Conceptually, we have moved from being focused on being interested in β, and more interested in improving the MSE of $\hat{Y}$
- The models can allow for a wide range of flexible functions
  - polynomials in underlying covariates, and interactions
- Will tend to do best when underlying model is not too non-linear
  - Also has more interpretable model parameters!
  - Today we’ll work on pushing this frontier

A primer on terminology

- One costly barrier to ML methods is a different terms for things
- Other terms worth knowing:
  - “one-hot encoding”: making dummy variables
  - “validation”: testing out of sample

Statistics	Machine Learning
Data Point	Instance
Covariate	Feature
Parameters	Weights
Estimation / Fitting	Learning
Regression / Classification	Supervised Learning
Clustering / Density Estimation	Unsupervised Learning
Response	Label
Test set performance	Generalization

Source: “Towards Data Science”

Distinctive Features: scalability + ensemble methods

- Many solution concepts for ML focus on the ability to
  1. Parallelize
  2. Split data up
  3. Recombine

- The parallization has obvious reasons: computational benefits

- Splitting the data up is less obvious, but pivots off of a notion of *averaging*
  - Effectively, randomly smaller samples (and random subsamples of features) will give a noisier estimate of the “truth”
  - However, averaging over many of these noisy estimates will quickly get at the right value

- This concept applies

Distinctive Features: Testing out of sample and cross-validation

- One big difference (although perhaps should not be) in ML is the use of out of sample testing to validate models
  - This matters a ton for ML models where there is subtantial overfitting in-sample

- Conceptually, consider a fully saturated model with many categorical variables on the right hand side:

$$Y_i = X_i\beta + \epsilon_i$$

  - β is the sample mean within each X_i bin
  - as $dim(X_i)$ grows relative to n, each bin more closely approximates exactly one observation
  - This will fit *very* well in-sample
    - It will be quite bad out of sample!

Distinctive Features: Testing out of sample and cross-validation

- Useful terminology: a training dataset, and a test dataset
  - Training: estimate the model
  - Test: evaluate the fitted model

- You can easily split a sample in this way by randomizing across observations
  - A crucial assumption here is independence of the observations (instances)
  - What if they're not independent? (particularly notable in asset pricing)
    - Can consider randomizing in blocks
    - Useful to have a theory that guides these decisions

What is supervised learning / ML?

- In essenece, we are interested in $f(X) = E(Y|X)$
  - This is exactly the same problem as our non-parametric regression question!
  - However, in that setting, I told you to give up as soon as $dim(X)$ got large because of data issues
    - ML folks are not so easily dissuaded!

- The solutions in ML revolve around different ways to circumvent a lack of data in higher dimensions
  - LASSO did this using penalized methods on global (linear) functions
  - Tree methods (next) do this by “widening” the comparable group
  - Deep learning-style methods “reduce” the dimensionality to improve comparability

Regression vs. Categorization

- Notably, ML methods distinguish between
  - continuous regression problems (e.g. price as an outcome)
  - classification problems (e.g. sold vs. not sold, or {red bus, blue bus, car})

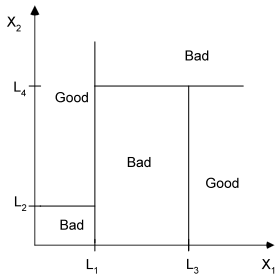
- In classification problems, the object is *not* necessarily to get the probability of an event (although this is doable)
  - The canonical example is digit recognition – the USPS needs to identify handwritten addresses
  - It doesn't care about an accurate measure of probabilities for each digit!

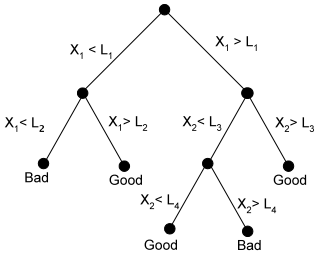
A brief tour through Trees and Forests

- A very popular off-the-shelf ML model that does both regression and classification is known as regression trees
  - Combinations of trees are called forests
- Given a set of parameter X_i with $dim(X) = k$, the goal is to take subsets of parameters, and split the sample based on values of individual covariates X_{ik}
- The choice of split is made to minimize within-sample error within each split sample
  - These splits are called “leaves”
- If we allowed infinite leaves, then the tree would grow until every observation was uniquely fit
  - The solution is to penalize the number of leaves allowed via “pruning”

Trees and Forests

- In essence this approach is like exhaustively dummying but:
  - The cutpoitns are chosen through the data
  - the interactions and cuts are penalized to avoid full saturation
- These models MASSIVELY overfit in sample, however, and so it is important to test out of sample!





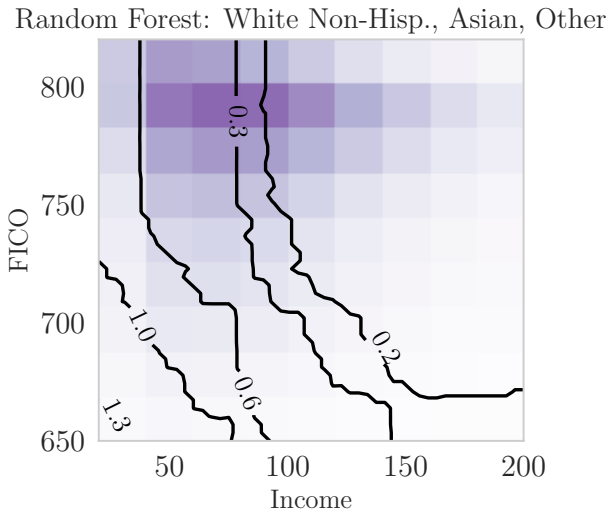
A brief tour through Trees and Forests

- These tree models are quite discrete in their cutpoints, which can have weird properties
- The traditional fix to this uses *Random Forests*.
- The approach makes many different trees to construct predictions, and then "averages" them. These trees differ in two ways:
  - They use random subsamples of the data
  - They use random subsamples of the covariates
- Random sampling creates independent variation across the predictions
  - Once averaged together, the overall predictions are quite a bit smoother and also better fit out of sample

An example Forest

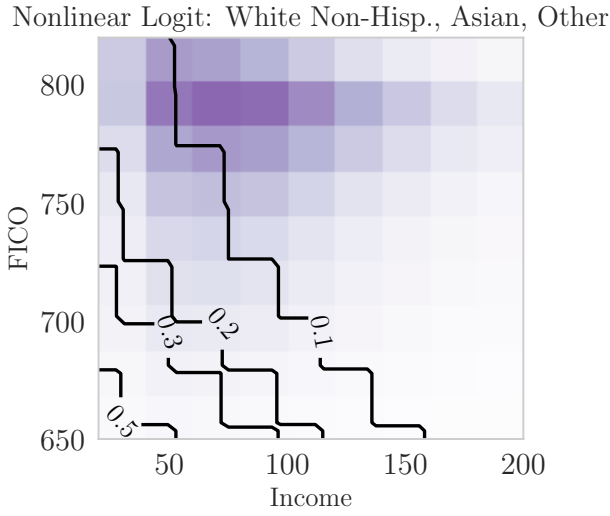
- These models can create much more non-linear relationships than say, Logit

- Example from Fuster et al. (2021)



An example Forest

- These models can create much more non-linear relationships than say, Logit
- Example from Fuster et al. (2021)



How to approach and further reading

- If you are considering ML models, the two standard programming languages with significant support are Python and R
  - It's possible that Julia and Matlab are doable, but I have seen far less here
- In Python, `scikit-learn` is a great package with many features
- In R, there is a big community of a packages that can be used as well, with `caret` as a very useful meta-engine for running diffferent estimations
  - "Hands-On Machine Learning in R" is an excellent resource for this: `https://bradleyboehmke.github.io/HOML/`
- The biggest challenge in my experience, is that the high-level concepts are very straightforward, while the nitty-gritty tuning, testing and decipering is quite hard
  - A big reason this is true is that it's very application-specific
  - Different forecasting problems have different issues

Checklist for what to look for

- What type of outcome do I have? (Classification vs. Regression)
- Do I expect highly non-linear responses, or relatively linear ones?
  - Many economics models have relatively monotonic response patterns
  - Linear models might do just fine!
- Do not use these models for data exploration – they can be computationally quite expensive, and are much less transparent
  - Identify your prediction problem carefully, and setup your estimation accordingly to cross-validate on appropriate parameters