Machine What Now?
Machine Learning in Political Science

Methods Lunch Talks

Sam Fuller[1]

1 Thanks to Chris Hare for the slides from which this presentation is based

Methods Talks

Welcome to the first methods lunch talk!

This will be a weekly lecture series featuring grad students (like myself) presenting on various methods and their applications.

The first few weeks, like this one, will feature general overviews of methods, whereas later this quarter we hope to feature applications of these methods on political science data.

We want to hear your feedback! Please stick around to let us know what you’d like to see from this series.

Overview

1. What is Machine Learning?
2. Supervised vs. Unsupervised ML
3. ML in Political Science
    - a Categories of Models
    - b Opportunities and Limitations

What do We Mean by ‘Learning’?

Machine vs. Statistical Learning

The primary goal of machine learning is **accurate prediction.**

It is highly related to the entire field of Artificial Intelligence (AI) and is concerned with large-scale applications (think Google's DeepMind) and both **in-sample** and, but more importantly, **out-of-sample** prediction accuracy.

This is in contrast to statistical learning which, like its name suggests, is more related to statistics with a larger emphasis on model-building, interpretability, and identifying **precision** and **uncertainty.**

Standard Statistical Models

Standard statistical models in political science have a *specified* functional form and parameters.

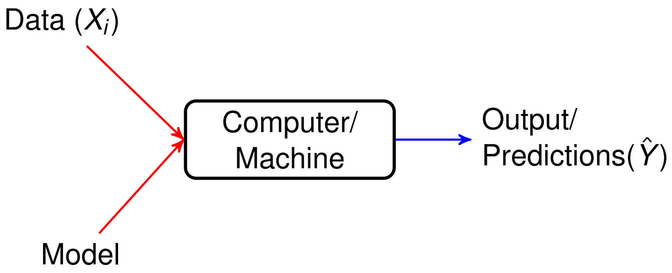
Let's consider, for example, a basic OLS equation with k variables:

$$Y = \beta_1 X_1 + ... + \beta_i X_k + \varepsilon_i$$

In this context, we decide what variables are included, any

transformations (e.g. logs, exponents). This model doesn't really 'learn' anything, it just solves the minimization problem of least squares. Everything is specified by us.

Standard Statistical Models



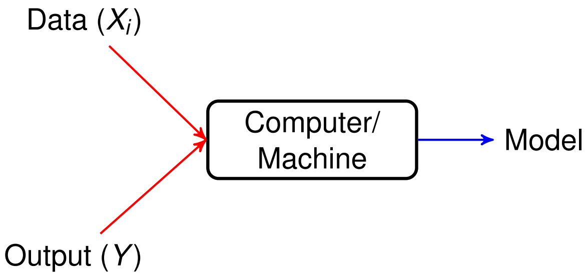
Machine Learning Models

Machine learning, on the other hand, does not make any assumptions about the equation/functional form to be used to run an analysis.

In fact, it takes in the data and tests a bunch of different models with different functional forms, interactions, etc. until it finds the model that comes up with the best predictions (in the supervised learning case).

This is where the **‘learning’** comes in: the algorithm determines the most predictive model by learning from the data (both Xs and Ys).

Machine Learning Models



Supervised vs. Unsupervised Learning

What is Supervision?

I have to come clean, that last figure is slightly misleading, and is actually representative of **Supervised** ML.

Supervised Machine Learning has an output that's specified, some value that it is trying to predict (Y) given a set of explanatory variables (X_i).

Unsupervised Machine Learning has *no* specified output only a set of explanatory variables (X_i). It attempts to find relationships between these variables, but is not attempting to predict any of them.

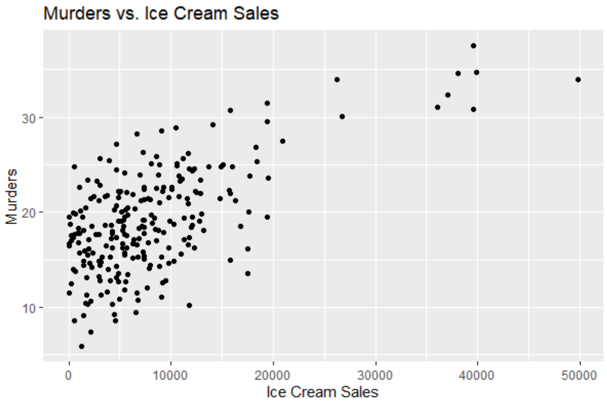
Supervised Learning

So with supervised learning, using both X_i and Y, we wish to:

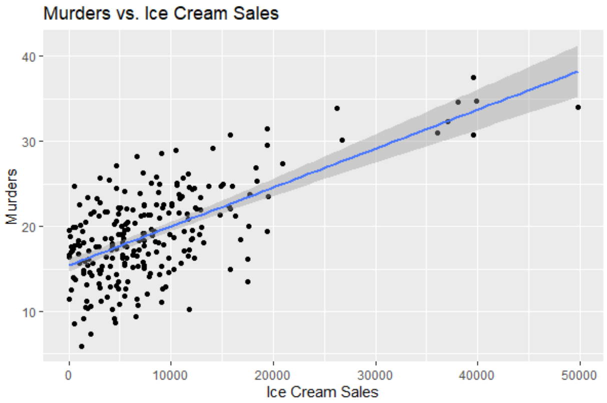
1. Understand which inputs affect the outcome, and how.
2. Assess the quality of our predictions and inferences.
3. Accurately predict unseen test cases.

Let's look at an example.

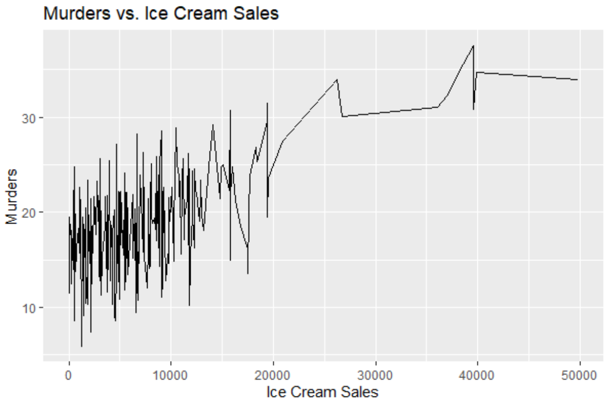
How Should We Model This?



With a Linear Model?



With a Perfectly Predictive Model?



Bias-Variance Tradeoff

These two cases highlight the fundamental bias-variance tradeoff:

1. We can have a relatively predictive model, that has some errors, and it would likely do ok with new data, this is the case of OLS.
2. We can also have a perfectly predictive model, that has no errors, but would likely do terribly with new data, this is the case of the line.

Because of the nature of ML algorithms, that they are attempting to maximize predictive power, we have to limit the possibility of overfitting.

Model Error

Model performance or fit (for any kind of model, not just ML) is assessed with the use of some **loss function**, which aggregates model error (ε), or the discrepancy/residuals between actual values of Y and predicted values of Y (expressed as $\hat{Y}$). That is:

$$\hat{Y} = \hat{f}(X)$$
$$\varepsilon = Y - \hat{Y}$$

The OLS loss function, for instance, looks to minimize the sum of squared errors (SSE): $\sum \varepsilon^2$ or $\sum(Y - \hat{Y})^2$. Machine learning methods often use mean squared error (MSE), or $\frac{SSE}{n}$.

Model Error

We need estimates of two kinds of model error to deal with the bias-variance tradeoff:

1. The error rate for the data used to estimate the model: how well does the model fit existing data?
2. The error rate for outside data: how well does the model fit new data?

Training vs. Testing Sets

Because ML cares so much about predictive accuracy, especially out of sample, we divide our data to get the error rates from the last slide.

Specifically, we divide our data into **training** and **testing** sets:

1. **The Training Set** is used to 'train' the model. It's the data the algorithm works with to create and test models itself.
2. **The Testing Set** is used to 'test' the model. It's data the algorithm cannot use and, consequently, helps test the generalizability of the model (how well the model applies to new data).

Measuring Predictive Accuracy

We use *both* the training and testing sets to measure the accuracy of our model, but we focus more on the testing set statistics. It sometimes the case where the model is very accurate on the training set, but much less so for the testing set (this can be sign of overfitting).

Cross-Validation and Tuning Parameters

To train the algorithm, we must first specify the criteria for which to measure the model. This can range from simple accuracy (% correct) to ROC (which measures both specificity and sensitivity). We must also often specify **tuning parameters** which change how the algorithm finds the best model.

Furthermore, we also use cross-validation in the algorithm. Specifically, instead of just measuring the predictive accuracy on the whole of the training set, it divides the training set itself into another training and testing set.

The standard in ML is to use k-fold cross validation.

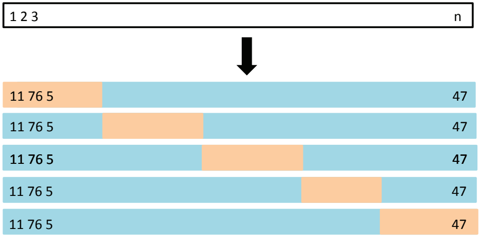
Cross-Validation

How do we estimate training and test error? A validation-set approach is one popular resampling technique:

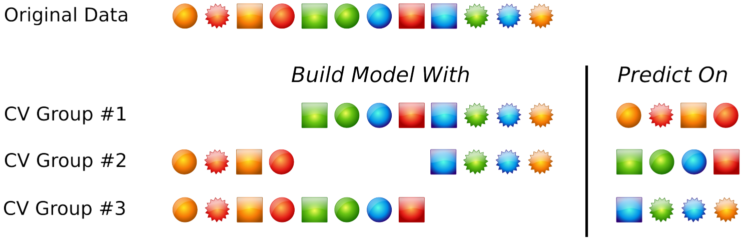
1. Randomly divide the available set of samples into two parts: a training set and a validation or hold-out set.
2. Estimate the model using the remaining of the data.
3. Apply the model to the observations in that subset, generating predictions ($\hat{Y}_{\text{test}} = \hat{f}(X_{\text{test}})$) and residuals ($Y_{\text{test}} - \hat{Y}_{\text{test}}$) to estimate testing error.

K-Fold Cross-Validation

- Of course, we hate to lose data when estimating the model.
- K-fold cross-validation offers one solution: randomly divide the data into K equal-sized parts. We leave out part k, fit the model to the other $K-1$ parts (combined), and then obtain predictions for the left-out kth part.
- This is done in turn for each part $k = 1, 2, \ldots, K$ and then the results are combined.



K-Fold Cross-Validation

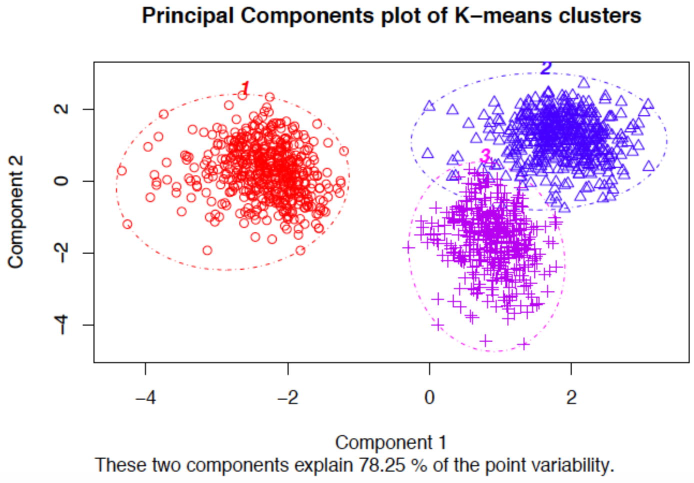


Unsupervised Learning

Fundamental components:

- No outcome variable, just X: a set of predictors (features) measured on a set of samples.
- Objective is more exploratory in nature: find groups of observations that behave similarly, find features that behave similarly, or find linear combinations of features with the most variation.

Cluster Analysis



ML in Political Science

ML Applications in Political Science

Standard social science modeling problems:

1. Variable selection
2. Dimensional reduction
3. Heterogenous treatment effects
4. Missing data imputation
5. Latent variable estimation
6. Explore causal relationships
7. Simulate counterfactuals
8. Estimate propensity scores

Supervised Problems

There are two broad categories of supervised learning problems:

- Classification: Similar to logit and probit, Y is categorical and we wish to correctly classify cases given a set of covariates.
- Regression: Similar to OLS, Y is continuous and we wish to accurately predict the value Y takes on given a set of covariates.

Examples of Models

There are many types of algorithms/models used in ML. Here are just a few:

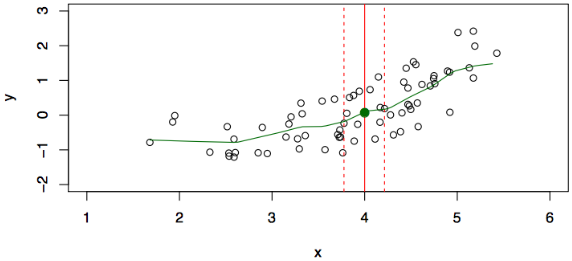
- k-nearest neighbors
- Random Forest
- Gradient Boosting Machines
- Neural Networks

Let’s take a look at one method called k-nearest neighbors.

Example: *k*-Nearest Neighbors

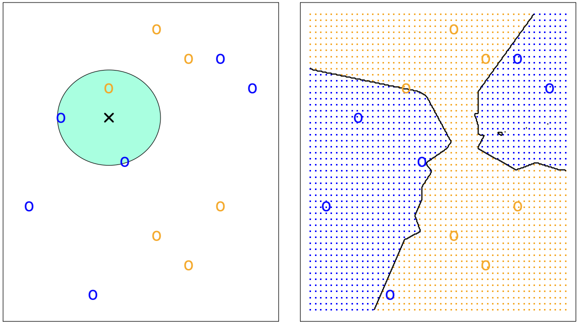
We wish to estimate f: $E(Y \mid X = x)$. Ideally, we would have lots of data point at every value of X. More realistically, with sparse data, we need to expand to include points in the **neighborhood** (ν) of a given value of X:

$$\hat{f}(x) = Ave(Y \mid X \in \nu(x))$$



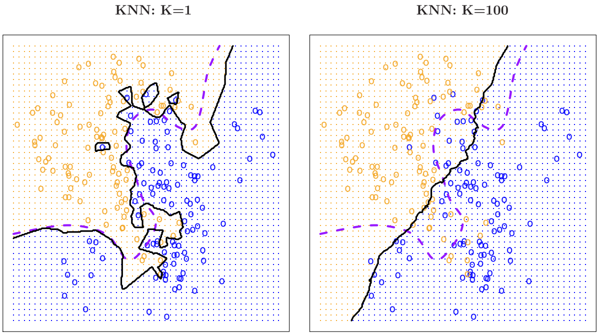
Example: *k*-Nearest Neighbors

With classification problems, we will predict x based on the classes of its k closest neighbors, extrapolating that process to map predictions for all points in the space:



Example: *k*-Nearest Neighbors

The choice of *k* determines the flexibility of the prediction surface:



Too Cold?

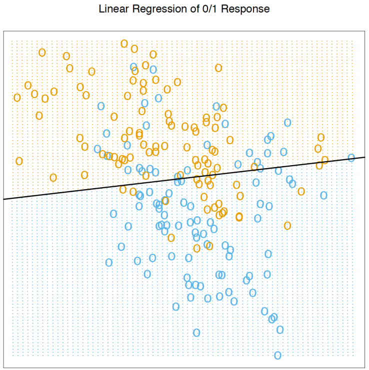


FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (*BLUE *$= 0$,* ORANGE *$= 1$), and then fit by linear regression. The line is the decision boundary defined by $x^T\hat{\beta} = 0.5$. The orange shaded region denotes that part of input space classified as* ORANGE*, while the blue region is classified as* BLUE.

Too Hot?

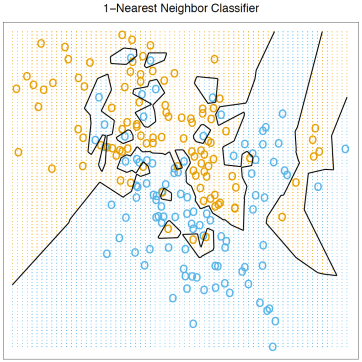


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$), *and then predicted by 1-nearest-neighbor classification.*

Just Right (k=15)

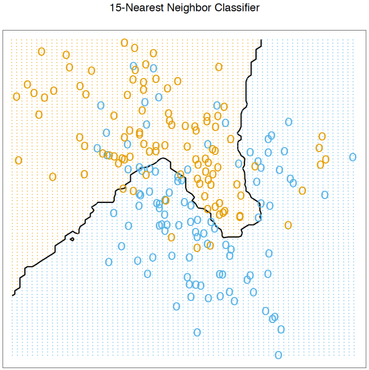


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.*

Unsupervised Models

There are also quite a few types of unsupervised methods too. In Political Science, these often have to do with dimensional reduction or classifying individuals:

- Principal Component Analysis
- Cluster Analysis

Really, this should be motivated by the curse of dimensionality.

The Curse of Dimensionality

- As p increases and the dimensionality $\uparrow$, the volume of the space explodes and the data points grow further and further apart.
- How many people in this room share the same birth month, as opposed to share the same birth month *and* height *and* party ID *and* . . .
- Sparse data means we have less leverage to estimate the effects of X on Y (that is, f).
- Frankly, this is probably better appreciated in the machine learning world and too often ignored by social scientists. In machine learning, this motivates dimensional reduction/feature selection methods.

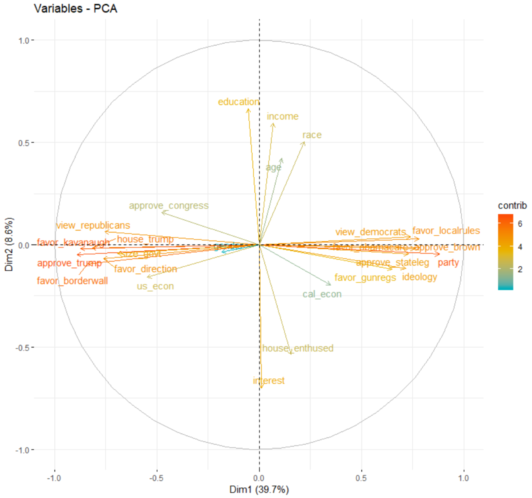
Example: Principal Component Analysis

In that vein, let’s take a look at a dimensional reduction example using Principal Component Analysis (PCA).

Let’s say we want to predict someone’s vote-choice in a congressional election. However, we want to see if there is some underlying ‘dimension’ that exists among the survey questions that we ask (e.g. policy preferences, party ID, demographic variables etc.).

PCA generates dimensions using the data that, in order, contain the most variance. Looking at the plot of the first few dimensions that PCA finds can be very illustrative. And we can actually use these identified dimensions in regular statistical models (like logit or probit).

PCA Plot


Variables - PCA
education
income
race
age
approve_congress
view_republicans
house_trump
favor_kavanaugh
size_govt
approve_trump
favor_direction
favor_borderwall
us_econ
view_democrats
favor_localrules
favor_obamacare
approve_brown
approve_stateleg
party
favor_gunregs
ideology
cal_econ
house_enthused
interest
contrib
6
4
2
Dim1 (39.7%)
Dim2 (8.6%)
-1.0
-0.5
0.0
0.5
1.0

Opportunities for Future Research

There are some really exciting opportunities for future research with ML on data that we have yet to really exploit quantitatively.

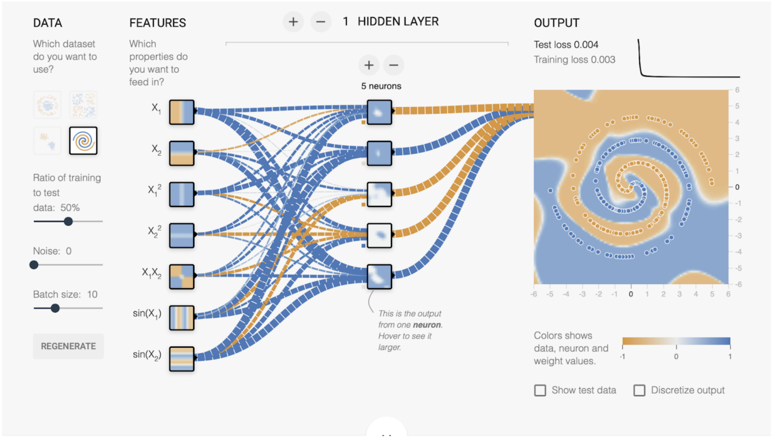
Namely image and text analysis are now widely possible due to neural networks.

Limitations

The biggest limitation with machine learning applications is causal inference and the interpretation of the (un)importance of variables.

In OLS, it is easy to see the importance of one variable, but in most ML circumstances it can be very hard to determine variable importance, or to understand the different transformations and interactions the model uses.

Neural Network


DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

X_1, X_2, X_1^2, X_2^2, X_1X_2, $\sin(X_1)$, $\sin(X_2)$

1 HIDDEN LAYER

5 neurons

This is the output from one ***neuron****. Hover to see it larger.*

OUTPUT

Test loss 0.004

Training loss 0.003

Colors shows data, neuron and weight values.

Show test data

Discretize output

Conclusion

Overall ML presents an interesting future avenue for political scientists. For prediction purposes and for calculating measures such as propensity scores, ML is often unrivaled.

Also, it has opened up new avenues for models that can analyze massive amounts of images and text.

However, there are limitations for determining causality and for the interpretability of different components of the model.

Questions?

- What do you want to see from these talks?
- Was this talk helpful?
- How about logistics or food/drinks?
- Anything Else?