Empirical Evaluation of Management Practices I

Predictions and Machine Learning

Prof. Dr. Dirk Sliwka Jesper Armouti-Hansen

17/11/20

Content

1. Introduction to Machine Learning

General definition

[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.

- (Arthur Samuel, 1959)

More specific definition

A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

- (Tom Mitchell, 1997)

Examples of ML/AI in business

- Employee attrition prediction
- Recruiting automation
- Customer churn modeling
- Recommendation engines

Types of Machine Learning

Supervised Learning:

The task of learning a function that maps the input(s) X to an output y based on example input-output pairs.

- Regression: The output is continuous or discrete and ordered.
  - Example: Predicting house prices based on house characteristics.
- Classification: The output is a discrete and unordered set.
  - Example: Classifying an email as spam or ham based on the use of certain words.

This is a "mini-course" on supervised learning with Python.

Unsupervised Learning:

We observe inputs but no output. We can seek to understand the relationship between the variables or the observations.

- For example, we might observe multiple characteristics for potential customers.
  - We can then try to cluster potential customers into groups based on these characteristics
- We might also try to project our inputs into a lower dimensional space.
  - This can be a beneficial preprocessing for supervised learning when dealing with high-dimensional data.

We will not deal with unsupervised learning in this course.

Terminology

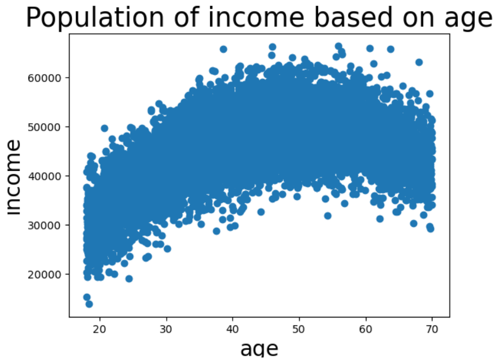
The terms used in the ML literature differs slightly from that used in Econometrics:

- Supervised learning $\rightarrow$ *Regression, classification, predicting* y_i *given* x_i.
- Features $\rightarrow$ x_i, *independent variables, explanatory variables, regressors, predictors.*
- Target $\rightarrow$ y_i, *dependent variable.*
- Training $\rightarrow$ *Estimating a model.*
- Testing $\rightarrow$ *Evaluating a model.*
- Training data $\rightarrow$ *The sample we use to train our model.*
- Test data $\rightarrow$ *The sample we use to test our model.*

2. Regression

Let us start with an example:

- Suppose our task is to predict income based on a person's age and that we had data on the whole population:



- What would be a "good" prediction here?

More formally

- Let X_i be a random vector (i.e. the vector of features).
- Let Y_i be a real variable (i.e. the response).
- We are interested in a function $f(X_i)$ which makes "good" prediction about Y_i.
- To know what a "good" prediction is, we require a loss function: $L(Y_i, f(X_i))$ which penalizes bad predictions
  - Common choice: Squared error – penalizes the quadratic distance:

$$L(Y_i, f(X_i)) = (Y_i - f(X_i))^2 \tag{1}$$

Recall from the lecture on Part 2:

CEF Prediction Property

Let $f(X_i)$ be any function of X_i. The conditional expectation function (CEF) solves:

$$E[Y_i|X_i] = \arg\min_{f(X_i)} E[(Y_i - f(X_i))^2] \tag{2}$$

- Thus, in terms of prediction, we can do no better than the CEF
- Also, recall:

CEF Decomposition Property

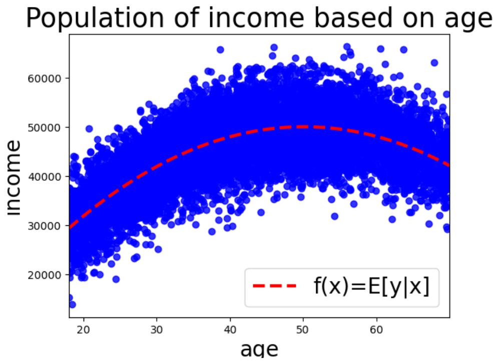
We can decompose Y_i such that

$$Y_i = E[Y_i|X_i] + \epsilon_i \tag{3}$$

Where:

1. ϵ_i is mean independent of X_i: $E[\epsilon_i|X_i] = 0$.
2. ϵ_i is uncorrelated with any function of X_i.

Thus, if we knew the population, calculating the CEF is usually not a difficult task:



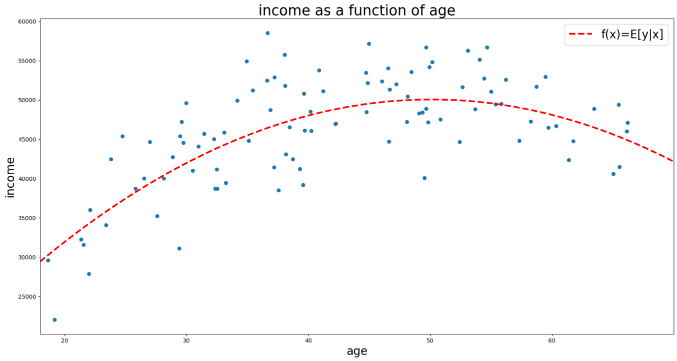
We just predict the average income for people of the same age.

- In this case, we are dealing with simulated data with the following conditional expectation function:

$$f(income) = 2000 * age - 20 * age^2 \qquad (4)$$

Usually, we do not know the whole population. Rather, we are working with a sample.

- In this example, we will be working with a sample of 100 data points.
- Our goal is to construct a model $\hat{f}(age)$ which makes "good" predictions about income



Estimating f - K Nearest Neighbor (KNN) Regression

The CEF (f) is almost always unknown, so how can we estimate it?

- For a given observation x_i, we could approximate the CEF by predicting the average of Y_i across observations with $X_i = x_i$.
- Problem?
  - We might have very few or no other observations with $X_i = x_i$
- Instead, we can settle with predicting the average of Y_i of the K nearest known neighbors of x_i:

$$\hat{f}(x_i) = \frac{1}{K} \sum_{j \in N_K} y_j \tag{5}$$

  - Where N_K is a neighborhood containing the indices of the K closest x's.

Estimating f - Linear Regression

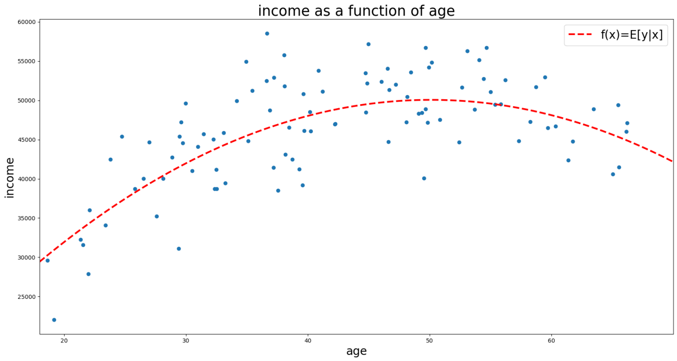
Another way to estimate the CEF is to assume that it is approximately linear in its arguments:

$$\hat{f}(x_i) = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \cdots + \hat{\beta}_k x_{ik} \tag{6}$$

- Thus, estimating the CEF amounts to finding the β's that minimize the squared error in the sample.
- *We now know of two machine learning models: KNN and Linear Regression!*

Example

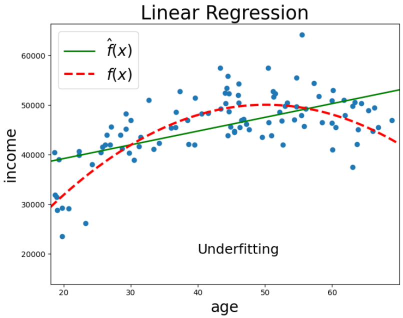
Let us return to our example:

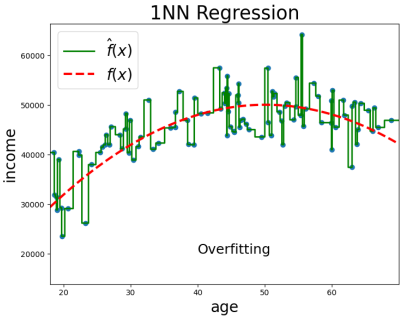


Underfitting vs. Overfitting

In order to make predictions based on our sample, suppose we train two models: A simple linear regression:

1. $\hat{f}(age_i) = \hat{\beta}_0 + \hat{\beta}_1 age_i$
2. A KNN regression with $K = 1$





Clearly, the linear regression is too inflexible and the 1NN regression is too flexible to properly approximate the CEF.

The simple linear regression makes bad predictions because it underfits our training data:

- It has high bias and low variance.

The 1NN regression makes bad predictions because it overfits our training data:

- It has low bias and high variance.

Thus, when dealing with predictions, we face a trade-off:

- The Bias-Variance trade-off

Essentially, we want to find a model that has low bias and low variance, if possible.

The Bias-Variance trade-off

More formally, given a realized point x_0, the expected squared error is given by:

$$E[(y_0 - \hat{f}(x_0))^2] = Var[\hat{f}(x_0)] + [f(x_0) - E[\hat{f}(x_0)]]^2 + \sigma_\epsilon^2 \qquad (7)$$

The first term is the variance of our model at x_0, the second the squared bias at x_0, and the third is the irreducible error.

Introduction to Sci-kit Learn (sklearn)

Linear Regression in sklearn

- To fit a linear regression model on the data X and y, we first import the module:
  `from sklearn.linear_model import LinearRegression`
- Then we can perform a regression with the following code:
  `reg = LinearRegression().fit(X,y)`
- We can access one of its attributes to get the coefficients:
  `reg.coef_`
- Suppose the regression is given by $\hat{f}(x_i) = \hat{\beta}_0 + \hat{\beta}_1 * age + \hat{\beta}_2 * age^2$ and that I would like to get the prediction for a 20 year old person:
  `reg.predict([[20, 20**2]])[0]`

KNN Regression in sklearn

- To fit a KNN regression model on the data X and y, we first import the module:
  `from sklearn.neighbors import KNeighborsRegressor`
- Then we can perform a regression with the following code:
  `knn = KneighborsRegressor().fit(X,y)`
- Again, suppose we want the prediction of a 20 year old person:
  `knn.predict([[20]])[0]`

Drawing a random sample with pandas

- Suppose we would like a random sample of 50 observations from a pandas dataframe `df`:
  `df_sample = df.sample(n=50, random_state=181)`

Your tasks

1. Import the income dataset as a pandas dataframe. It is located at https://raw.githubusercontent.com/armoutihansen/EEMP2020/main/datasets/income.csv
2. Create a new column in the dataframe "age_sq" that takes the squared values of the age variable.
3. Since the CEF can now be estimated by a linear regression, draw a random sample of 100 observations and fit a linear regression on the sample. Are the coefficients close to those given by the CEF?
4. Write a loop that generates a new sample 100 times of 100 observations and fit a linear regression on each of the 100 samples. Store the coefficients of the model in each loop.
5. Calculate the mean of each coefficient. Are the means close to the coefficients given in the CEF?

Your tasks*

1. Consider the observation *age* $= 50$. What is the conditional expectation of income at that age?
2. Drop the "age_sq" variable from the dataframe and write a loop that generates a new sample 100 times of 100 observations. On each sample, fit a 1NN regression and store its income prediction of a person aged 50 in each sample.
3. Calculate the mean of the predictions and plot the distribution of the predictions. What does this tell you about the 1NN's bias and variance on this dataset?

3. Classification

Let us start with an example:

- Suppose our task is to predict whether a person defaults on her debt based on her income and creadit card balance. Again, suppose we had data on the whole population:

IMAGE

- What would be a good prediction here?

More formally

Suppose there are B classes. We wish to estimate a "good" classifier $f(X_i)$ that assigns a class label to any observation X_i.

- To know what a "good" prediction is, we require a loss function: $L(Y_i, f(X_i))$ which penalizes bad predictions.
  - Common choice: Misclassification rate:

$$L(Y_i, f(X_i)) = \begin{cases} 1 & \text{if } f(X_i) \neq Y_i \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Bayes' Optimal Classifier (BOC)

- Let $p_b(x_i) = Pr(Y_i = b|X_i = x_i), b = 1, \dots, B$
- Suppose we knew this conditional probability.
- Then the _Bayes' optimal classifier at any x_i is given by:

$$f(x_i) = b \quad \text{iff} \quad p_b(x_i) = \max\{p_1(x_i), \dots, p_B(x_i)\} \tag{9}$$

- In other words, for any observation, predict that it belongs to its most likely class.
- Notice that we are utilizing the conditional probability distribution in a similar way to how we utilized the conditional expectation function for regression problems.

- If we knew the population, calculating the BOC is often not a difficult task:
- IMAGE

We calculate the condional probability of defaulting at any point and then predict the class with the highest probability.

- The dotted line is called the decision boundary.

Usually, we do not know the whole population. Rather, we are working with a sample.

- In this example, we will be working with a sample of 1000 data points.
- Our goal is to construct a classifier $\hat{f}(income, balance)$ which makes "good" predictions about whether a person defaults.
- IMAGE

Estimating f - K Nearest Neighbor (KNN) Classifier

The BOC (f) is almost always unknown, so how can we estimate it?

- We could approximate the conditional probabilities by calculating the relative frequency for all B classes for x_i.
- Then we predict the class with the highest approximate conditiona probability.
- Problem?
    - We might have very few or no other observations with $X_i = x_i$.

- Instead, we can settle with estimating the conditional probabilities $p_b(x_i), b = 1, \dots, B$ using the K nearest known neighbors of x_i:

$$\hat{p}_b(x_i) = \frac{1}{K} \sum_{j \in N_K} y_j \tag{10}$$

  - Where N_K is a neighborhood containing the indices of the K closest x's.

- Finally, we predict that x_i belongs to the class with the highest estimated conditional probability:

$$\hat{f}(x_i) = b \quad \text{iff} \quad \hat{p}_b(x_i) = \max\{\hat{p}_1(x_i), \dots, \hat{p}_B(x_i)\} \tag{11}$$

Estimating f - Linear Regression

If $B = 2$, we can convert the classes into 0's and 1's and perform linear regression to estimate the conditional probabilities.

- For example, *default* $= 1$ and *no default* $= 0$.

Then we can treat the output of the linear regression as the conditional probability of the class that has been converted to 1's.

- For example, $\hat{p}_{default}(x_i) = \hat{\beta}_0 + \hat{\beta}_1 * income_i + \hat{\beta}_2 * balance_i$

Finally, we classify x_i to the class with the highest estimated conditional probability.

- For example,

$$\hat{f}(x_i) = \begin{cases} \text{default} & \text{if } \hat{p}_{default}(x_i) > 0.5 \\ \text{no default} & \text{otherwise} \end{cases} \tag{12}$$

Note that that there are some drawbacks with using linear regression for classification:

- We might get conditional probability estimates below 0 and above 1.
- When B$>$2, linear regression imposes cardinality assumptions on the classes.

Because of these points, other models are often preferred such as logistic regression.

Example

Let us return to our example:

- IMAGE

Underfitting vs. Overfitting

In order to make predictions based on our sample, suppose we train two models:

1. A linear regression: $\hat{p}_{default}(x_i) = \hat{\beta}_0 + \hat{\beta}_1 * income_i + \hat{\beta}_2 * balance_i$
2. A KNN regression with $K = 1$

- IMAGE

Clearly, the linear regression is too inflexible and the 1NN regression is too flexible to properly approximate the BOC.

Logistic Regression

Logistic regression is a classification method that is more appropriate than linear regression when there are more than two classes and tends to perform better than KNN regression with high-dimensional data.

- Here we introduce the method applied to our example, but the intuition holds in the general case with more than two classes.
- To avoid probabilities outside $[0, 1]$, we estimate $p_{default}(x_i)$ using the sigmoid function:
- IMAGE

- Applying the sigmoid function implies estimating the conditional probability as follows:

$$\hat{p}_{default}(x_i) = \frac{exp(\hat{\beta}_0 + \hat{\beta}_1 * income_i + \hat{\beta}_2 * balance_i)}{1 + exp(\hat{\beta}_0 + \hat{\beta}_1 * income_i + \hat{\beta}_2 * balance_i)} \quad (13)$$

- We then classify x_i as default if this estimated conditional probability is larger than 0.5.
- Note that the decision boundary is the set of points: $\{x_i|\hat{p}_{default} = 0.5\}$ which is equivalent to $\{x_i|\hat{\beta}_0 + \hat{\beta}_1 * income_i + \hat{\beta}_2 * balance_i = 0\}$.

Example

- IMAGE

Introduction to Sci-kit Learn (sklearn)

KNN Classification in sklearn

- To fit a KNN Classifier on the data X and y, we first import the module:
  `from sklearn.neighbors import KNeighborsClassfier`
- Then we can perform a classification with the following code:
  `clf = KNeighborsClassifier(n_neighbors=n).fit(X,y)`
- We can access one of its attributes to get the coefficients:
  `reg.coef_`

Your tasks

1. Import the income dataset as a pandas dataframe. It is located at https://raw.githubusercontent.com/armoutihansen/EEMP2020/main/datasets/income.csv
2. Create a new column in the dataframe "age_sq" that takes the squared values of the age variable.
3. Since the CEF can now be estimated by a linear regression, draw a random sample of 100 observations and fit a linear regression on the sample. Are the coefficients close to those given by the CEF?
4. Write a loop that generates a new sample 100 times of 100 observations and fit a linear regression on each of the 100 samples. Store the coefficients of the model in each loop.
5. Calculate the mean of each coefficient. Are the means close to the coefficients given in the CEF?

Your tasks*

1. Consider the observation $age = 50$. What is the conditional expectation of income at that age?
2. Drop the "age_sq" variable from the dataframe and write a loop that generates a new sample 100 times of 100 observations. On each sample, fit a 1NN regression and store its income prediction of a person aged 50 in each sample.
3. Calculate the mean of the predictions and plot the distribution of the predictions. What does this tell you about the 1NN's bias and variance on this dataset?

4. Model Selection and Assesment

Assessment of the general performance (or general error) of our predictive model is what we truly care about. The general performance of a model refers to its prediction ability on independent test data.

- Or, more generally, its predictive capability on the population.

A model's capability on the training data is often a biased estimate of its general performance.

- Why?

If we can estimate models' general performance, we:

1. Can select the optimal model for our problem;
2. Know how well this optimal model can predict on the population.

Types of errors

Note: We will consider the regression setting here in which the performance is evaluated using the squared error. However, the intuition also holds for the classification setting.

- The *training* error of a model $\hat{f}$ trained on a *training* sample T is simply the mean squared error in that sample:

$$\bar{err} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}(x_i))^2 \tag{14}$$

- The *test* error is the expected squared error of our model for a new observation drawn from the population, conditional on being trained on the training sample T:

$$Err_T = E[(Y_0 - \hat{f}(X_0))^2 | T] \tag{15}$$

- Finally, the *expected test* error is the expected error for a new observation over everything that is random - including the training sample T:

$$Err = E[E[(Y_0 - \hat{f}(X_0))^2 | T]] = E[(Y_0 - \hat{f}(X_0))^2] \qquad (16)$$