Economics 882

James G. MacKinnon

January, 2025

Introduction

- ECON 882 is a masters-level course that deals with **machine learning**, also called **statistical learning**.
- It is assumed that all students have taken a serious masters-level econometrics course like ECON 852.
- The emphasis will be on developing an intuitive understanding of the key ideas of statistical learning, as well as familiarity with a number of widely-used methods.
- Students are expected to learn how to employ a variety of machine-learning methods using either R or Python. The instructor is more familiar with R.
- Lectures: Tuesdays 10:00–11:20, Thursdays 8:30–9:50.
- There will also be tutorials.

- Much of the course will be based on the books
  - *An Introduction to Statistical Learning, with Applications in R*, Second Edition, by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani (Springer, 2021); hereafter ISLR.
  - *An Introduction to Statistical Learning, with Applications in Python*, by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor (Springer, 2023); hereafter ISLP.

  These are essentially the same book, except for the computer programs. They can legally be obtained as PDF files.
- The course may also cover material based on the more advanced book *Elements of Statistical Learning* by Trevor Hastie, Robert Tibshirani, and Jerome Friedman, Second Edition (Springer, 2009)—hereafter ESL— and on other sources.
- In addition, some use will be made of the book *Applied Causal Inference Powered by ML and AI* by Victor Chernozhukov, Christian Hansen, Nathan Kallus, Martin Spindler, and Vasilis Syrgkanis, arXiv:2403.02467; see `causalml-book.org`.

- The R language is free and open-source. It can be installed from the website `https://cran.r-project.org/`
- R users will need to install a number of libraries, such as "glmnet" and "ISLR2". This can be done over time. See Section 2.3 of ISLR.
- The Python language is free and open-source. For installation instructions, see Section 2.3 of ISLP.
- In R, a library can be installed using, for example,
  - install.packages("glmnet")

  Once a library been installed on a particular computer, it remains there forever. It can then be used by entering the command
  - library(glmnet)

  for every R session or script file where it is to be used.
- Unfortunately, installing R packages can be challenging, especially on Linux and Mac systems. I suggest using the `r2u` system on the former.

- The course website is here:

  `http://qed.econ.queensu.ca/pub/faculty/mackinnon/econ882/`
- Regular assignments (probably 3 of them) will be worth 40% of the final mark, and the empirical project will be worth 60%.
- Students should choose their own topic and dataset for the empirical project. It would be advisable to discuss the choice with the instructor and/or the TA.
- Answers to each assignment should be submitted to the TA as a single PDF file.
- The due date for the empirical assignment will depend on when grades have to be submitted to the Graduate School. It will probably be around the very end of April.

Types of Statistical Learning

The course will focus on **supervised learning**, where one variable, the **output**, is treated differently from the others.

The objective is usually to predict the output variable. Supervised learning involves estimating a sort of **reduced form**.

Perhaps we have 223,741 observations on house values, along with observations on many features of each house. Then the value (or its logarithm) is the output, and the other features are **inputs**.

When there is no output variable in the dataset, we have to use **unsupervised learning**.

We might have thousands of pictures of animals, but nothing to indicate which animals they are. The object might be to group them into animals of the same species, or sets of related species.

Cluster analysis is an unsupervised method used for exploratory data analysis to find hidden patterns or groupings.

Principal components analysis, or **PCA**, is a form of unsupervised learning that is widely used in econometrics.

Of course, PCA can also be used as part of a supervised learning exercise when there are many explanatory variables.

Sometimes, machine learning methods can generate their own data.

For example, **generative adversarial networks**, or **GANs**, are machine-learning methods in which two neural networks play games against each other.

A **generative network** generates candidate datasets, and a **discriminative network** evaluates them.

GANs can be used to generate fake photographs and videos that look stunningly realistic.

Supervised Learning

The inputs, of which there are often many, are sometimes called **predictors** or **features**. Economists might call them **explanatory variables** or **regressors**.

The output, of which there is often just one, is sometimes called the **outcome** or the **response**. Economists might call it the **dependent variable** or **regressand**.

Some outputs are quantitative, and often approximately continuous, in which case the prediction task is usually called **regression**.

Some outputs are **categorical** or qualitative, in which case the prediction task is usually called **classification**. The possible output values are often called **classes**.

Some machine/statistical learning methods are designed for regression, and some are designed for classification.

Many have two variants, one for regression and one for classification.

The distinction between regression and classification is not hard and fast. Both linear and nonlinear regression can be used for classification.

- When asked to make a prediction, statistical-learning methods for classification typically just pick one class.
- However, it might be more informative to assign probabilities to each of the classes.
- Simply choosing the class with the highest probability is rarely optimal, especially when some mistakes are costlier than others.
- For example, a self-driving car may have to decide whether it is about to hit a deer or a moose.

Some ML methods are designed for a small number of inputs, which are allowed to affect the outcomes in a very general way.

Others are designed for a large number of inputs, which are often allowed to affect the outcomes in very restrictive ways. Many of them will ultimately be discarded, or given very little weight.

Some methods can handle both small and large numbers of inputs.

ISLR/ISLP uses n for the number of observations (at least, the number used for estimation), and p for the number of inputs.

Sometimes N is used nstead of n. Econometricians often use k or K to mean p, or often $p + 1$, since p does not count the constant term.

Methods for small numbers of inputs include **kernel regression** and **smoothing splines**. There is a large econometric literature on non-parametric regression, often based on kernel methods. See

> *Henderson, D. J., and C. F. Parmeter (2015). Applied Nonparametric Econometrics, Cambridge University Press.*

Methods for large numbers of inputs are called **high-dimensional** methods. The best known of these is the **lasso**.

Closely related methods are **ridge regression** and **elastic net** regression.

Such methods can handle problems with far more inputs than observations. For example, $n = 1000$ with $p = 200,000$ can be feasible!

The sample used for estimation is called the **training dataset**, or **training sample**. In econometrics, this is usually the entire sample.

A fundamental problem with all prediction methods is **over-fitting**. It is very likely to occur unless we penalize models that fit too well.

This happens to some extent with any statistical learning method, but it tends to be very severe with flexible methods.

It is therefore common to hold back part(s) of the sample.

A **validation dataset**, or **validation sample**, may be used to prevent over-fitting. The training sample is used for estimation, but the validation sample is used to determine certain **tuning parameters**.

- For example, both lasso and ridge regression require a tuning parameter, often denoted λ. The larger λ is, the more the coefficients are shrunk to (or towards) zero.
- Kernel regression involves a **bandwidth** parameter. The larger it is, the smoother is the estimated regression function.

The trick is to find the value of the tuning parameter that yields the best predictions for the validation sample.

Instead of holding back a validation sample, **cross-validation** can be used. The training sample is divided up in a number of different ways.

- With **leave-one-out cross-validation**, there are n subsamples, each with $n-1$ observations. Estimates based on each subsample are used to predict the omitted observation.
- With 10-fold cross-validation, the training sample is divided into ten 90/10 subsample pairs. The estimates from each of the 90% subsamples are used to predict the associated 10% subsamples.
- 5-fold cross-validation, which involves dividing the sample into five 80/20 pairs, is also popular.
- A number of values of the tuning parameter are tried for each of the validation sample pairs. The one that yields the best predictions is (usually) then used with the training sample.

- Some modern methods use **cross-fitting**, where both estimation and validation are done on parts of the sample.
- The final estimates ultimately use the entire sample, but estimation is never done on the entire training set at once.
- In the 5-fold case, we estimate 5 different models, one for each of the 80/20 sample pairs.
- Then we average the resulting predictions somehow.

Whether or not there is a validation sample, it is common when trying to see how well one or more methods work to reserve some data for a **test dataset** or **test sample**, or **testing sample**.

Usually, this is done at random, and care should be taken to ensure that the split really is random.

Depending on the nature of the sample, this can be tricky to do. If some observations may be correlated with others, we do not want to put correlated observations into different subsamples.

For time-series data, dividing the data randomly makes no sense.

Instead, we need to divide the sample at a certain point in time. Earlier data are used for training, and later data are used for testing.

If data come in as time passes, a test sample may naturally accumulate after the original training/validation sample was gathered.

For example, we may be trying to predict what fraction of patients recover from some condition, perhaps as a function of the treatments they are given.

If new patients are constantly being added to our sample, we probably don't need to hold back part of the dataset for testing, because many new cases will come along.

Some Key Concepts

A nonlinear regression can be written as

$$y_i = f(\boldsymbol{x}_i) + u_i, \quad i = 1, \ldots, n. \tag{1}$$

Here $\boldsymbol{x}_i$ is $p \times 1$, and u_i is assumed to have $\mathrm{E}(u_i) = 0$.

Econometricians typically assume that the functional form of $f(\cdot)$ is known, although its parameters are not.

In particular, the **linear regression model** assumes that

$$f(\boldsymbol{x}_i) = \boldsymbol{x}_i^\top \boldsymbol{\beta} = \boldsymbol{X}_i \boldsymbol{\beta}, \tag{2}$$

where $\boldsymbol{\beta}$ is a p-vector of coefficients to be estimated, and $\boldsymbol{X}_i = \boldsymbol{x}_i^\top$ is a row vector of observations on the inputs.

Some statistical-learning methods assume that (2) holds, but a great many do not. Most make weak assumptions about $f(\cdot)$.

One important issue is what the model (1) is intended for. Do we care about **inference**, **prediction**, or both?

If we believe that a linear regression model is correctly specified, inference is usually quite easy. We care about $\partial \mathrm{E}(y)/\partial x_k = \beta_k$.

To make inferences about β_k, we simply obtain a covariance matrix for $\hat{\boldsymbol{\beta}}$ and use it to construct a confidence interval for β_k.

This is often just $\big[\hat{\beta}_k - c_{1-\alpha/2}\, s(\hat{\beta}_k),\ \ \hat{\beta}_k + c_{1-\alpha/2}\, s(\hat{\beta}_k)\big]$, where $s(\hat{\beta}_k)$ is a valid standard error.

Inference is a bit harder for nonlinear regression models, because $\partial \mathrm{E}(y)/\partial x_k$ depends on $\boldsymbol{x}$ as well as $\boldsymbol{\beta}$.

The restrictive features of linear (and parametric nonlinear) regression models that make inference easy can make prediction not work well.

In contrast, machine-learning models are generally designed to make good predictions, with little or no thought to inference.

For such models, inference can be challenging.

Prediction always involves errors. For any predicted observation, there are two types of error: the **reducible error** and the **irreducible error**.

Suppose that $y = f(\boldsymbol{x}) + u$. Then, even if we knew $f(\boldsymbol{x})$, both its functional form and all its parameters, the squared error would be

$$\mathrm{E}\big(y - f(\boldsymbol{x})\big)^2 = \mathrm{E}(u^2) = \mathrm{Var}(u). \tag{3}$$

This squared error is *irreducible*. We cannot do any better.

The expectation of the actual squared error for the test sample is

$$\mathrm{E}\big(y - \hat{f}(\boldsymbol{x})\big)^2 = \mathrm{E}\big(f(\boldsymbol{x}) + u - \hat{f}(\boldsymbol{x})\big)^2 = \mathrm{E}\big(f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})\big)^2 + \mathrm{Var}(u). \tag{4}$$

The first term in the last expression is the *reducible squared error*.

The second equation in (4) depends on the fact that the randomness in $\hat{f}(\boldsymbol{x})$ comes solely from randomness in the training sample and is therefore independent of u in the test sample.

The reducible squared error arises from two sources. Suppose the true model is $g(\boldsymbol{\theta}, \boldsymbol{x})$, we know it, and we obtain estimates $\hat{\boldsymbol{\theta}}$. Then

$$\mathrm{E}\big(f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})\big)^2 = \mathrm{E}\big(g(\boldsymbol{\theta}_0, \boldsymbol{x}) - g(\hat{\boldsymbol{\theta}}, \boldsymbol{x})\big)^2. \tag{5}$$

Even though we know f here, the squared error may be large if $\hat{\boldsymbol{\theta}}$ is a poor estimator of $\boldsymbol{\theta}_0$.

More realistically, the $g(\cdot)$ model we estimate is usually wrong. Then

$$\mathrm{E}\big(f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})\big)^2 = \mathrm{E}\big(f(\boldsymbol{x}) - g(\boldsymbol{\theta}_0, \boldsymbol{x})\big)^2 + \mathrm{E}\big(g(\boldsymbol{\theta}_0, \boldsymbol{x}) - g(\hat{\boldsymbol{\theta}}, \boldsymbol{x})\big)^2, \tag{6}$$

where $\boldsymbol{\theta}_0$ is the value of $\boldsymbol{\theta}$ that makes $g(\boldsymbol{\theta}, \boldsymbol{x})$ as close as possible to $f(\boldsymbol{x})$.

We can reduce the first term by making $g(\boldsymbol{\theta}, \boldsymbol{x})$ provide a better approximation to $f(\boldsymbol{x})$, thus reducing the **bias**.

We can reduce the second term by reducing the variance or (more generally) the mean squared error of $\hat{\boldsymbol{\theta}}$.

But these two ways of reducing squared error often work at cross-purposes.

Parametric models typically involve a modest number of parameters (unless p is large). Very flexible non-parametric methods typically involve many parameters.

More parameters give a model greater flexibility to approximate complicated, nonlinear $f(\boldsymbol{x})$ functions. Thus, increasing the number of parameters should make the first term in (6) smaller.

But estimating a lot of parameters accurately typically requires a very large sample. Otherwise, the second term in (6) may be large.

A **smoothing parameter** or **tuning parameter** (or perhaps more than one such parameter) often determines how flexible a model is.

There is typically a **bias-variance tradeoff**. With little **flexibility** (a lot of smoothness), the bias is large but the variance is small.

With little or no **smoothness** (i.e., a lot of flexibility), the bias is small, but the variance is large.

See Chapter 2 of ISLR/ISLP.