Machine Learning Methods for Econometrics
Winter 2025

Stéphane Bonhomme

University of Chicago

- **Introduction: Machine learning and economics**
- **Linear regression**
- **Penalized Regression**
- **Factor methods**
- **Nonlinear models**
- **Partitioning methods**
- **Re-sampling methods**

Schedule

1. Week 1A: Introduction
2. Week 1B: Linear regression I
3. Week 2A: Linear regression II
4. Week 2B: Penalized regression I
5. Week 3A: Penalized regression II
6. Week 3B: Penalized regression III
7. Week 4A: Factor methods and matrix completion I
8. Week 4B: Factor methods and matrix completion II
9. Week 5A: **Projects: first presentation**
10. Week 5B: Nonlinear models and neural networks I
11. Week 6A: Nonlinear models and neural networks II
12. Week 6B: Nonlinear models and neural networks III
13. Week 7A: Partitioning methods and trees I
14. Week 7B: Partitioning methods and trees II
15. Week 8A: Resampling methods and random forests I
16. Week 8B: Resampling methods and random forests II
17. Week 9A: Review session and questions
18. Week 9B: **Projects: final presentation**

Introduction: Machine Learning and Economics

1. **What is Machine Learning?**
2. **Examples in Economics**
3. **Outline of the course and references**

What is Machine Learning?

Machine Learning can be understood as statistical engineering.

The "engineering" solutions where developed to address various new problems (ex: speech recognition) and analyze new type of data (ex: text data).

But ML borrows a lot from statistics. Some "fancy" ML techniques (re-sampling, cross-validation, even stochastic gradient descent) were actually invented a long time ago by statisticians.

Lots of ideas are well-known in statistics, although the terminology differs between stats and ML.

See Tibhirani's glossary: https://statweb.stanford.edu/ tibs/stat315a/glossary.pdf

Now statistics (and its econ branch, econometrics) are immensely useful to economists, both in micro and macro.

The questions is thus: are the new techniques developed in ML for certain types of problems useful to solve economic problems?

To think about this, it is useful to review several key ideas behind the ML view of the world:

- A key challenge for ML is the curse of dimensionality. It is always present in statistics, but it is particularly acute in ML applications. The challenge is both statistical and computational.Bot the number of data points and the number of parameters increase (and standard errors can still be large with big data).
- A common solution to the curse of dimensionality is regularization. The idea is to modify the estimator to make it well-behaved in some (though not all) circumstances. Penalization ideas.
- ML researchers have a clear metric of success, which they use to compare different methods. The metric is out of sample prediction. The idea is that methods typically

overfit in sample, especially if they depend on many parameters. This is relevant in ML since the type of estimators that we will see typically have many parameters.

A standard recommendation is to split the sample into 3 parts:

estimation/tuning/evaluation

The metric of success is important when one thinks about the relevance of ML for economics. ML typically focuses on short-run predictions, while economists care about predictions in the long run, or at least quite far way from the available data.

- Not often emphasized in ML textbook, but key for the success of ML methods, are the properties of the underlying data generating process. At an abstract level, these methods require some form of sparsity.

  Example of a linear regression:

$$y_i = \sum_{j=1}^{p} \beta_j x_{ij} + u_i,$$

  where only some of the β_j's are different from zero. In that case, some form of penalized regression may correctly recover (some of) the β_j's.

  We hope to exploit sparsity, or "structure". If data is too complex there are sharp limits on what we can learn.

- ML methods often involve a lot of computation.

  Some of the key innovations in ML involve finding "tricks" to speed up computation or make it numerically more reliable. The 2024 Nobel Prize to Geoff Hinton is an illustration of this focus.

- ML's approach is algorithmic. Leo Breiman distinguishes "two cultures" in statistics:

  -The model-based culture, traditional in statistics.

  -The black-box, algorithmic culture, which dominates in ML.

  Now, it is important to assess how successful results based on the black-box approach are. If one disposes of a clear assessment of success (such as OOS performance) then a model may not be needed. However, in most applications models (and economic "priors") are needed given the type of data at our disposal and the ambitious questions we are interested in.

- ML is a part of statistics where model selection plays a central role.

  A simple example is to select the set of relevant regressors in a sparse regression model, i.e., which β_j's are equal to zero and which ones are different from zero?

  Estimating the model may seem a strange idea, but it is becoming increasingly popular given the growing size and quality of data sets.

- An example of model selection is for regression on a basis of functions. For example, regress an outcome on polynomial functions of a covariate (x, x^2, x^3)...

  One possibility is to estimate β_k the model

$$y_i = \sum_{k=0}^{K} \beta_k \phi_k(x_i) + u_i,$$

  where $\phi_k(x_i) = x_i^k$.

  Another possibility is to select the basis that is most relevant to explain y. That is, using parametric functions $\phi_k(x_i; \theta_k)$ that depends on parameters θ_k, one can estimate β_k and θ_k in the model

$$y_i = \sum_{k=0}^{K} \beta_k \phi_k(x_i; \theta_k) + u_i.$$

  Hence, one chooses the relevant basis $\phi_k(x_i; \theta_k)$ in a data-driven way. This is a key feature of most ML methods (trees, neural networks, kmeans...).

- Model averaging is another common feature of many ML methods.

  For example, one can average over possible estimates of β_j (e.g., obtained by selecting s out of the p regressors). A key question is how to weight the terms in the average.

  Relative to model selection, model averaging (and "ensemble" methods) tend to work better for prediction and forecasting tasks.

- You may be familiar with simulation-based methods, Monte Carlo simulation, and the bootstrap. These stochastic methods are extremely common tools in ML.

  This includes re-sampling (e.g., random forests), Monte Carlo, Bootstrap, split sampling, bagging...

After introducing some of the key concepts we will see at play in the course, which characterize ML methods, it is useful to ask three "high-level" questions and think about them before we delve into details during the next few weeks.

- If black-box ML methods are dominant, is there still a role for statistical theory?

  ML methods are notoriously difficult to analyze theoretically. There are important challenges when trying to establish consistency of estimators, risk guarantees, and validity of inference.

  In particular, the use of model selection raises important challenges challenges when trying to compute confidence intervals that are valid from a classical (frequentist) statistical perspective. This is the problem of "inference post model selection" that we will discuss later in the course.

  Irrespective of the theoretical difficulties, trying to specify concrete assumptions about the data generating process (DGP or "true model") is still immensely helpful. One can then check how various ML methods perform in such a DGP, or "laboratory". When analysis based on pen and pencil is too hard one can always resort to Monte Carlo simulations. We will make extensive use of MC simulations in the course.

- Isn't ML all about prediction? Is there still a role for causal thinking?

  The answer is a resounding "yes". While causation versus prediction is a central theme of data interpretation, many causal methods require solving some prediction tasks as building blocks.

  For example: predict the outcome given the treatment of interest and covariates, and predict the treatment given covariates, and then combin the two into a "doubly robust" estimator of the average treatment effect.

  There is recent influential work on treatment effects methods using ML (by authors such as Athey, Imbens, Wager, Chernozhukov, C. Hansen...).

- Finally, if ML methods are automatic black boxes, is there still a role for economic models?

  Absolutely! Economic models can provide plausible specifications for DGPs to be used to assess the performance of ML methods in economic settings.

  In addition, there is growing work trying to marry ML methods with the computation and estimation of structural economic models (in particular in macroeconomics, as in Fernandez-Villaverde's work).

Examples in and outside Economics

Outside economics: many successes. Ex: chess (new algorithms and new computers), pattern and speech recognition, text analysis (latent variable models and "denoising", transformers – the T in ChatGPT)...

Inside economics: not dominant yet, but growing:

- Demand analysis: many attributes, many products.
- Dynamic games: large state space.
- Network models.
- New use of text data (Gentzkow/Shapiro and others).
- Econometric literature on Lasso, and then double-debiased ML (increasingly popular).
- Record linkage for matching data sets.
- Estimating average treatment effects under selection on observables (Athey/Imbens).
- Estimating structural models via flexible indirect inference (Kaji/Manresa/Pouliot).

Outline of the course and references

- Linear regression: many covariates.
- Penalized regression and the Lasso.
- Factor methods.
- Methods for neural networks and other nonlinear models
- Partioning: trees and k-means clustering
- Re-sampling methods: the bootstrap, bagging and random forests

References:

- Hastie, Tibshirani and Friedman: Elements of Statistical Learning.
- James, Witten, Hastie and Tibshirani: An Introduction to Statistical Learning with Applications in R.

- Murphy: Machine Learning, a probabilistic perspective.
- For theory: Hastie, Tibshirani Wainwright on sparsity and the Lasso.
- Articles in econometrics, statistics and machine learning/computer science journals.

Chapter 1: Linear Regression

1. **Low-dimensional regression: reminder**
2. **Why many covariates?**
3. **Challenges with many covariates**
4. **Selecting covariates: economic and automatic methods**

Low-dimensional regression: reminder

OLS: definition

Model.

$$y_i = \sum_{j=1}^{p} \beta_j x_{ij} + u_i, \quad i = 1, ..., N.$$

Matrix form:

$$y = X\beta + u.$$

X has dimensions $N \times p$.

OLS estimator. $\widehat{\beta} = (X'X)^{-1}X'y$.

Best linear predictor for sample square loss. Mimics the population counterpart: best linear predictor (BLP) for expected square loss.

OLS: finite-sample and asymptotic properties

Behavior under normality.

$$u \,|\, X \sim \mathcal{N}(0, \sigma^2 I),$$

where the identity matrix I has dimension p.

Then:

$$\widehat{\beta} \,|\, X \sim \mathcal{N}(\beta, \sigma^2 (X'X)^{-1}).$$

Can be used to form exact confidence intervals about β in a classical statistical setting.

Consistency. Standard asymptotic analysis: N tends to infinity, p fixed.

Important remark: asymptotic analysis is a fiction, only useful in so far as it approximates well the finite sample behavior.

Assumptions: (1) $\mathbb{E}(uX) = 0$, (2) $X'X/N$ tends in probability to a non-singular matrix, (3) $u'X/N$ tends in probability to zero.

Note that (1) amounts to defining β as the population BLP. In many settings, the BLP is not the coefficient of interest since the economic unobservable u (e.g., "ability") is correlated with the economic observable X (e.g., years of education). In those cases OLS does not recover the parameter of interest, even in large samples.

Use the law of large numbers and the continuous mapping theorem (under suitable conditions) to show $\widehat{\beta}$ tends to β in probability as the sample size N tends to infinity.

Asymptotic distribution. If observations are i.i.d. we use the central limit theorem (CLT) to show that, as N tends to infinity:

$$\sqrt{N}(\widehat{\beta} - \beta) \xrightarrow{d} \mathcal{N}(0, \mathbb{E}[x_i x_i']^{-1}\mathbb{E}[u_i^2 x_i x_i']\mathbb{E}[x_i x_i']^{-1}),$$

where x_i is $p \times 1$.

Note: $X'X = \sum_{i=1}^{N} x_i x_i'$. It is very useful to learn to go back and forth between different types of expressions, involving X, x_i, and x_{ij}.

Remark on variance estimation: White's (1980) formula and its extensions (notably, clustered-White). With independent observations the White formula is:

$$(X'X)^{-1}X'\widehat{\Omega}X(X'X)^{-1},$$

where $\widehat{\Omega}$ is diagonal with elements $\widehat{u}_i = (y_i - x_i'\widehat{\beta})^2$.

Notice the presence of $(X'X)^{-1}$ in this expression.

Coding: OLS estimator and its variance.

Why many covariates?

Many predictors. Economic example I: economic growth, in a panel of countries. More theories and potential growth determinants than data points. See Sala i Martin (1997).

Economic example II: macro/finance forecasting. Stock and Watson's series of papers.

Should we include all regressors at the same time or only a subset, or alternatively a (linear or nonlinear) combination of them?

Many instruments. In instrumental variable regression, it is tempting to add instruments in order to increase th R^2 in the first stage. Often, researchers interact an original instrument z_i with other covariates w_{ij} and use $z_i w_{ij}$ as instruments.

Recently, it has become popular to use "judge fixed-effects" as instruments for judges' decisions in different economic settings (disability insurance, crime, ...). This also gives rises to a large number of instruments.

Nonparametric regression: series estimation.

$$y_i = f(x_i) + u_i.$$

Here x_i has a small dimension. For example: x_i is scalar.

Approximate $f(x)$ in a basis of functions:

$$f(x) \approx \sum_{k=0}^{K} a_k \phi_k(x),$$

where $\phi_k(x)$ may be (1) ordinary polynomials, (2) Chebyshev, Legendre, Hermite... polynomials, (3) splines...

Here we need to let K grow to approximate f well.

The approximation is typically of the form:

$$\left|\left| f - \sum_{k=0}^{K} a_k \phi_k \right|\right| \leq C K^{-\alpha},$$

for some constants $C > 0$, $\alpha > 0$, and a norm $||\cdot||$ on functions.

Examples of norms of functions: ℓ^1 norm $||f||_1 = \int |f(x)| dx$, ℓ^2 norm $||f||_2 = (\int |f(x)|^2 dx)^{1/2}$, sup norm $||f||_\infty = \sup_x |f(x)|$, ...

Challenges with many covariates

Large X matrix: instability

Exact and approximate collinearity. Key question: is $X'X$ exactly or approximately collinear?

Checking collinearity on the computer. (1) Compute the rank of $X'X$ (numerically), (2) compute the eigenvalues of $X'X$.

Exact collinearity is very rare (but it happens! A classical example is the indeterminacy of age, time, and cohort effects in a linear regression). However: approximate

multicollinearity is very frequent in applications.

Question: if X_1 and X_2 have a correlation of .99 should we still treat them as not collinear?

Important remark: (approximate) multicollinearity becomes more prevalent as the number of covariates increases.

Singular value decomposition (SVD). $X = UDV'$. U, V have orthogonal columns, D is diagonal, with dimensions equal to the rank of X.

We can order the diagonal elements of D in decreasing order. The key is the minimum element. Indeed:

$$(X'X)^{-1} = VD^{-2}V'.$$

Key issue: very often, when p is moderately large (say 20, 30) the minimum element of D is close to zero.

Remark: the SVD is a very useful tool when working with matrices more generally.

Coding: Try different (large) random matrices and compute their SVD.

Many instruments: caution! When using too many instruments relative to the sample size, the instrumental variables estimator (two-stage least squares – 2SLS) tends to be biased. The bias is towards the OLS estimator. In such cases there is overfit in the first stage: z_i predicts x_i "too well", since the model used to predict x_i is too flexible given the sample size.

Classic references are Angrist and Krueger (1991) and Bound, Baker and Jaeger (1995, JASA).

Overfit is a theme that we will return to many times in this class.

Large p: need for a different theory

Standard asymptotic analysis. In a standard asymptotic analysis, p is fixed as N tends to infinity. In the large-N limit we thus learn β very well in general. This is because the estimation noise becomes very small when N is large.

Asymptotically, even $p = 500$ is small relative to a (very very) large N.

This approach may have unrealistic consequences. For example, OLS should work well according to this even with a very large number of covariates (say, $p = 500$), but often it does not.

Alternative asymptotic: the "growing p" trick. The trick is to let p increase as the sample size N increases. We let $p = p_N$ be a function of N.

This means that the population is changing with the sample size! In fact we are now considering a sequence of populations, indexed by N.

This approach captures finite sample performance better in cases where p is not negligibly small relative to N.

Formally, in this approach the rate of growth $p_N = p(N)$ is key: the faster p_N grows with N the more difficult the estimation problem.

Remark: what a given rate p_N means for any given application is sometimes unclear. In practice we just have one N and one p. However, the alternative asymptotic setup can motivate new estimators that tend to work better in applications. It is also very useful to construct confidence intervals that have a good coverage.

OLS under large p: what do we know?

Some results with large p. The rate of p_N as a function of N is key to ensure good performance of OLS.

In a classic paper, Huber (1973) shows that consistency and asymptotic normality of $\widehat{\beta}$ require p/N tending to zero.

Recently, Cattaneo, Jansson and Newey (2018) focus on one coefficient of β, say β_k, and show that consistency and asymptotic normality can still be achieved for such a one-dimensional parameter even when p/N tends to a non-zero constant.

Another case that has been widely studied is series estimation.

Series estimators. Rates of convergence and asymptotic normality. Newey (1997, Journal of Econometrics).

An important result of the theory is a characterization of the ℓ^2 convergence rate:

$$\left|\left|\sum_{k=0}^{K} \widehat{a}_k \phi_K - f_0\right|\right|_2^2,$$

where the coefficients $\widehat{a}_0$, ..., $\widehat{a}_K$ are simply OLS estimates when regressing y_i on $\phi_0(x_i)$, ..., $\phi_K(x_i)$.

The square bias is proportional to $K^{-2\alpha}$, for some α that is larger the "smoother" the true function f_0 is. The smoothness of a function measures how many terms ϕ_k are needed to approximate the function well.

The variance is proportional to K/N. This makes intuitive sense: we are estimating K parameters with N data points. Note that in the low-dimensional case (that is, under

the standard OLS theory) the variance is proportional to $1/N$ since K is a constant in that case.

Bias-variance trade-off. Bias(K) decreasing, Variance(K) increasing.

This trade-off motivates selecting K "not too large, not too small". The theory provides some guidance on K: if the ℓ^2 rate is the desired objective, it should be selected to be proportional to $N^{1/(2\alpha+1)}$, so as to balance squared bias and variance.

However, the proportionality constant is unknown (and the "smoothness constant" α is also typically unknown). Selecting K in practice remains a challenging problem.

A popular approach is to rely on cross-validation.

Selecting covariates: economic and automatic methods

Economic priors

An example: Mincer regression (role of theory). In a wage regression, there is a very large number of potential controls (occupation, tenure/seniority, ...).

The Mincer model of human capital motivates a parsimonious specification based on education and a polynomial in experience.

Another example: credit scores (role of economic environment). Suppose we want to predict the fact that an individual is given access to credit (for a mortgage, say). The data may contain an extremely large number of potential covariates to include (such as the full history of employment and wages, similar information for the spouse, parents and children of the individual, gender, race, ...).

However, it may be that for legal reasons the credit agency only uses a small subset of these variables when deciding to grant credit access or not.

In such cases, not including these "irrelevant" covariates may be very helpful. In this course we will see methods that provide data-driven ways of removing such irrelevant covariates. However, if one has knowledge, based on economic theory or institutional settings, one should use it, possibly in combination with the data-driven methods.

Appeal and limitations of subset selection

Subset selection. From the theory and simulations we learn that OLS is often badly behaved in the presence of many covariates. Econometrics and machine learning offer

a menu of alternative methods that may perform much better than OLS under suitable conditions, even when p is large.

Subset selection is perhaps the simplest method of all.

Let m be the maximum number of covariates that we want to include in our regression.

Here we have in mind a "small" m. We write $m << p$.

In subset selection, we compare the R^2 coefficients in any possible regressions with m covariates. The set of covariates with the highest R^2 is the one we select.

Remark: choosing m is analogous to choosing K in series.

Problem: subset selection is a combinatorial numerical problem (even too hard for "big computers"). The Lasso approximates the solution to this problem, yet it is much simpler computationally.

Coding: How to implement subset selection?

Does subset selection "work"? It is useful to consider a simple example where one wishes to select one covariate out of p.

In this case subset selection amounts to selecting the covariate x_{ij} that has the highest correlation with y_i.

If we assume that the error term u_i is $\mathcal{N}(0, \sigma^2)$ then it is possible to show that subset selection will select the "right" covariate, as N tends to infinity.

A large number of machine learning methods are based on such model selection ideas.

Stepwise methods. There exist some alternatives to exhaustive, combinatorial search. Heuristic methods include stepwise forward regression, for example (see chapter 3.3 in Hastie, Tibshirani and Friedman's book). They are rarely used in econ.

Chapter 2: Penalized Regression

1. **The Lasso: intuition and implementation**

2. **Other penalization schemes**

3. **Economic applications of the Lasso**

The Lasso: intuition and implementation

The Lasso

The Lasso penalty. Replace the least squares minimization by:

$$\min_{b_1,\ldots,b_p} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} b_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |b_j|.$$

Read Tibshirani (1996), who originally proposed the Lasso.

In matrix form:

$$\min_{b} \|y - Xb\|_2^2 + \lambda \|b\|_1.$$

This is OLS + ℓ^1 penalty.

When $\lambda = 0$, the Lasso solution coincides with OLS.

When λ tends to infinity all $\widehat{\beta}_j$ tend to zero.

Norms in $\mathbb{R}^p$. ℓ^2, ℓ^1, ℓ_∞. Some inequalities.

Why sparse solutions? It is useful to consider the dual representation of the Lasso problem:

$$\min_{b} \|y - Xb\|_2^2, \quad \text{s.t. } \|b\|_1 \leq C.$$

There is a one-to-one mapping between C and λ. λ may be interpreted as the Lagrange multiplier associated with the constraint $\|b\|_1 \leq C$.

This representation gives insight about why the Lasso solutions tend to have a lot of zeros, i.e. t be "sparse".

The level curves of the OLS objective are ellipsoids. The constraints set is diamond-shaped. Like in standard utility maximization theory, corner solutions are likely.

In two dimensions (b_1, b_2), corner solutions correspond to either $b_1 = 0$ or $b_2 = 0$.

Computation

The univariate case. Consider the case where $p = 1$ and x is univariate.

In this case the solution of the Lasso is available in closed-form.

Indeed, we want to minimize:

$$\min_{b_1,...,b_p} \sum_{i=1}^{N} (y_i - bx_i)^2 + \lambda|b|.$$

We can plot the objective function as a function of b.

The solution is:

$$\widehat{\beta}^{Lasso} = sign(\widehat{\beta}^{OLS})(|\widehat{\beta}^{OLS}| - \lambda)^+.$$

Hence $\widehat{\beta}^{Lasso}$ and $\widehat{\beta}^{OLS}$ have the same sign.

The magnitude of $|\widehat{\beta}^{Lasso}|$ is always smaller than the magnitude of $|\widehat{\beta}^{OLS}|$. We say that the Lasso "shrinks" the OLS estimate towards zero.

When $|\widehat{\beta}^{OLS}| \leq \lambda$, $\widehat{\beta}^{Lasso} = 0$. We say that the Lasso "thresholds out" the small values of $|\widehat{\beta}^{OLS}|$. Hence λ can be interpreted as a threshold.

The Lasso estimator is a "soft thresholding" estimator, since it thresholds out some values and shrinks the other ones. This is to be contrasted with "hard thresholding", such as estimators based on the ℓ^0 penalty (see below).

The orthogonal case. Consider the case where there are p covariates, which are all orthogonal to each other.

$$\sum_{i=1}^{N} x_{ij}x_{i\ell} = 0 \quad \text{for all } j \neq \ell.$$

In that case the "long regression" of y_i on x_{i1},...x_{ip} is identical to p univariate regressions of y_i on x_{i1}, ..., y_i on x_{ip}.

Likewise, in the orthogonal case the Lasso estimator:

$$(\widehat{\beta}_1^{Lasso}, ..., \widehat{\beta}_p^{Lasso}) = \min_{b_1,...,b_p} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} b_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |b_j|$$

is given by p univariate Lasso solutions:

$$\widehat{\beta}_j^{Lasso} = sign(\widehat{\beta}_j^{OLS})(|\widehat{\beta}_j^{OLS}| - \lambda)^+, \quad j = 1, ..., p.$$

Should we penalize all regressors? In applications it is unclear we want to penalize all covariates.

For example, some key regressors (education in a wage regression, price in a demand equation) should always be included and penalizing them makes little sense.

Also, in fixed-effects regressions penalizing the fixed effects may not be appealing.

It is straightforward to modify the Lasso for this purpose:

$$\min_{b_1,...,b_p} \sum_{i=1}^{N} \left(y_i - \sum_{k=1}^{m} c_k w_{ik} - \sum_{j=1}^{p} b_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |b_j|.$$

To implement this, simply compute residuals of y_i, x_{i1}, ..., x_{ip} on w_{i1}, ..., w_{im}, call them y_i^{res}, x_{i1}^{res}, ..., x_{ip}^{res}, and compute:

$$\min_{b_1,...,b_p} \sum_{i=1}^{N} \left(y_i^{res} - \sum_{j=1}^{p} b_j x_{ij}^{res} \right)^2 + \lambda \sum_{j=1}^{p} |b_j|.$$

This is simply the Lasso estimator based on residuals. The equivalence between the two problems comes from the partitioned regression (Frisch-Waugh) theorem.

The Lasso objective is convex. Convexity of the Lasso objective function is obvious in the univariate case.

Note that the Lasso objective is not continuously differentiable: there is non-differentiability at zero.

In the multivariate case, the Lasso objective is convex as well. This is very useful, as there are no multiple local solutions, and we can reach the global minimum of the function.

Caution: standardizing covariates. The Lasso estimator is sensitive to the scaling of the covariates. A popular approach is to standardize all of the x_{ij} so that $\sum_{i=1}^{N} x_{ij} = 0$ and $\sum_{i=1}^{N} x_{ij}^2 = 1$.

Cyclic coordinate descent. The idea of cyclic coordinate descent is to update each b_j one at a time. Given all other parameters except b_j, the update of b_j is simply based on the objective:

$$\min_{b_j} \sum_{i=1}^{N} \left(y_i - b_j x_{ij} - \sum_{\ell=1}^{p} \mathbf{1}\{\ell \neq j\} b_\ell x_{i\ell} \right)^2 + \lambda |b_j|.$$

This is a univariate Lasso problem!

Since $\sum_{i=1}^N x_{ij}^2 = 1$, the solution is:

$$b_j = sign(\widehat{b}_j)(|\widehat{b}_j| - \lambda)^+,$$

where $\widehat{b}_j = \sum_{i=1}^N x_{ij}(y_i - \sum_{\ell=1}^p \mathbf{1}\{\ell \neq j\} b_\ell x_{i\ell})$.

We implement this method iteratively, starting from initial values of $b_1, ..., b_p$ and looping over $j = 1, ..., p, 1, ..., p, ...$ until numerical convergence.

There are other popular methods to estimate the Lasso, such as the LARS (least angle regression) algorithm.

Coding: cyclic coordinate descent for lasso.

Theoretical guidance

Here we ask the question: what does the Lasso estimate? (and does it estimate it well?)

A sparse Data Generating Process. Consider a model where the true parameters β_{j0} are such that at most s of them are zero, and all other β_{j0}'s are zero. The model is given by:

$$y_i = \sum_{j=1}^p \beta_{j0} x_{ij} + u_i,$$

where u_i is independent of $x_{i1}, ..., x_{ip}$ and normal with mean zero and variance one.

The Lasso recovers $\beta_{10}, ..., \beta_{p0}$ "quite well", under suitable conditions, provided s is "small".

This motivates the Lasso as the estimator of choice in sparse regression models.

Rates of convergence. To be more formal, let $\widehat{\beta}$ be the Lasso estimator and β_0 be the vector of true parameters. Both $\widehat{\beta}$ and β_0 are $p \times 1$ vectors.

A classical result in the literature (due to Bickel, Ritov and Tsybakov) gives conditions under which:

$$\|\widehat{\beta} - \beta_0\|_2^2 = \sum_{j=1}^p (\widehat{\beta}_j - \beta_{j0})^2$$

tends to zero in probability as N and p tend to infinity while s is kept fixed.

The conditions for this result to hold include the assumption that p/N does not tend to infinity "too fast". Yet it is important to note that the result allows p to grow to infinity faster than N.

Recovering the set of relevant covariates? Does the fact that $\|\widehat{\beta} - \beta_0\|_2^2$ tend to zero asymptotically imply that Lasso is going to "correctly guess" which coefficients β_{j0} are zero?

Not necessarily.

In fact, the literature provides conditions under which the Lasso consistently estimates the set of zero coefficients (and by implication the set of non-zero coefficients).

However, these conditions are quite stronger than the ones needed for the consistency in ℓ^2 norm.

Intuitively, it can be that the Lasso estimator incorrectly estimates β_j to be non-zero, while the true β_{j0} is in fact zero, but that the consequence of this "mistake" is not severe for $\|\widehat{\beta} - \beta_0\|_2^2$ since the Lasso estimates $\widehat{\beta}_j$ is small.

The fact that in general the Lasso makes "mistakes" about has important implications for how to compute standard errors for the Lasso.

What is a good λ? λ plays a familiar role of striking the balance between bias and variance. This is similar to other penalization/regularization problems.

When λ is close to zero $\widehat{\beta}^{Lasso}$ has a large variance.

When λ is very large the variance is reduced, but $\widehat{\beta}^{Lasso}$ is biased (towards zero).

Choice of λ: formula and cross-validation. The theory in Bickel, Ritov and Tsybakov justify the following formula for λ:

$$\lambda = A\sigma\sqrt{\frac{\ln p}{N}},$$

for some constant $A > 2\sqrt{2}$.

σ is unknown. One can attempt to estimate it as the RMSE in a regression with "many" covariates. However, it is not obvious how to do this reliably.

In addition, which A to choose is not obvious.

An alternative that is widely used is cross-validation.

Coding. CV for Lasso.

Standard errors: naive approach. How to compute standard errors for $\widehat{\beta}^{Lasso}$?

A naive approach is to treat the estimated zeros as if they were true zeros.

That is, treat the covariates that Lasso estimates do not have an effect on the outcome as absent from the regression.

Then, compute usual OLS standard errors (using the formula under homoskedasticity, or the White formula under heteroskedasticity), in that regression with s regressors.

In practice one could proceed in two steps: (1) recover the relevant regressors using the Lasso, (2) run OLS on the set of relevant regressors. This approach is referred to as "post-Lasso".

Standard errors: issues with the naive approach. The naive approach is problematic since it overlooks the fact that, if $\widehat{\beta}_j = 0$ this does not necessarily mean that $\beta_{j0} = 0$.

In particular, if I am interested in a coefficient c of w_i, and include controls $x_{i1}, ..., x_{ip}$ in the Lasso regression, it may be that the Lasso "misses" some of the relevant covariates.

This creates omitted variable bias for my estimate of c.

This issue has motivated the development of variants of the Lasso that tries to include additional covariates, such as the "double Lasso" method of Belloni, Chernozhukov, and Hansen (2014).

Other penalization schemes

A general penalized OLS formulation is:

$$\min_{b_1,...,b_p} \sum_{i=1}^{N} (y_i - \sum_{j=1}^{p} b_j x_{ij})^2 + \lambda Pen(b_1, ..., b_p).$$

There are many different penalty functions in the literature. They correspond to different ways of enforcing sparsity in the estimates.

In the next chapter we will see that similar ideas can be useful in matrix settings.

ℓ^0 penalization

Changing the penalty. Consider the problem:

$$\min_{b_1,...,b_p} \sum_{i=1}^{N} (y_i - \sum_{j=1}^{p} b_j x_{ij})^2 + \lambda \sum_{j=1}^{p} \mathbf{1}\{b_j \neq 0\}.$$

The ℓ^0 penalty differs from the ℓ^1 penalty since:

$$\|b\|_0 = \sum_{j=1}^{p} \mathbf{1}\{b_j \neq 0\}$$

only takes into account the number of non-zero b's ("extensive margin"), and discards the magnitudes of the coefficients ("intensive margin").

Link to information criteria (Akaike). The objective function is closely related to the AIC (Akaike) information criterion:

$$\min_{b_1,\dots,b_p} \sum_{i=1}^{N}(y_i - \sum_{j=1}^{p} b_j x_{ij})^2 + 2\sigma^2 \sum_{j=1}^{p} \mathbf{1}\{b_j \neq 0\},$$

where $\sum_{j=1}^{p} \mathbf{1}\{b_j \neq 0\}$ is the number of parameters that we wish to estimate.

Here the criterion is the value of the minimum.

In practice, computing this criterion requires setting a value for σ^2, as usual when setting a penalty value.

The AIC is widely used for model selection. It can be used for more general objective functions such as likelihood, generalized method of moments (GMM), ...

The Bayesian information criterion (BIC) is a related, influential information criterion. It is justified differently.

Combinatorics. Minimizing:

$$\|y - Xb\|_2^2 + \lambda \|b\|_0$$

is numerically hard.

This is because, unlike Lasso, the objective function is highly non-convex.

In non-convex settings, descent algorithms are often trapped in local optima. Starting the algorithm from several initial parameter values helps, but in high-dimensional settings the number of possible starting values is very large, and it is hard to assess whether we have reached the global optimum.

A possibility here is to fix the number of non-zero coefficients $\|b\|_0 = s$ and to minimize:

$$\|y - Xb\|_2^2, \quad \text{s.t. } \|b\|_0 \leq s.$$

This is subset selection.

We know that the optimization here is combinatorial, hence not practical.

The Lasso approximates the ℓ^0 problem, using a convex objective function that can be minimized easily.

Orthogonal regressors. Consider the dual formulation:

$$\|y - Xb\|_2^2, \quad \text{s.t. } \|b\|_0 \leq s.$$

As in Lasso, the solution has an explicit form in the univariate case:

$$\widehat{\beta}_j^{\ell^0} = \widehat{\beta}_j^{OLS}(|\widehat{\beta}_j^{OLS}| \geq |\widehat{\beta}_{(s)}^{OLS}|),$$

where $\widehat{\beta}_{(s)}^{OLS}$ is the s-th largest OLS coefficient.

However, due to non-convexity of the objective function, this observation is not useful to find the minimum of the function in the general case.

ℓ^2 penalization: Ridge regression

In Ridge regression, we use as a penalty the sum of the squares $\sum_{j=1}^p b_j^2$.

The Ridge objective.

$$\min_{b_1,\ldots,b_p} \sum_{i=1}^N \left(y_i - \sum_{j=1}^p b_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p b_j^2.$$

This is a convex function. Strictly convex when $\lambda > 0$.

Dual formulation. In this case the dual is:

$$\min_{b_1,\ldots,b_p} \sum_{i=1}^N \left(y_i - \sum_{j=1}^p b_j x_{ij} \right)^2, \quad \text{s.t. } \sum_{j=1}^p b_j^2 \leq C.$$

The level curves of the OLS objective are ellipsoids, and the constraint set is a ball.

Hence, unlike Lasso Ridge regression does not produce sparse solutions in the sense that typically all Ridge estimates will be non-zero.

Orthogonal covariates. When regressors are orthogonal to each other the Ridge regression estimator as a closed-form solution:

$$\widehat{\beta}_j^{Ridge} = \frac{\widehat{\beta}_j^{OLS}}{1+\lambda}, \quad j = 1, ..., p.$$

Matrix formulation and solution. More generally, we can write Ridge as minimizing:

$$\|y - Xb\|^2 + \lambda\|b\|^2.$$

The solution is:

$$\widehat{\beta}^{Ridge} = (X'X + \lambda I)^{-1}X'y,$$

where I is $p \times p$.

Notice that, even if $X'X$ is singular, $X'X + \lambda I$ is non-singular.

A SVD formulation. Let $X = UDV'$ be the SVD of X.

Here we write the "complete' SVD, so U is $N \times N$, D is $N \times p$ and V is $p \times p$.

We have:

$$\begin{aligned}(X'X + \lambda I)^{-1}X'y &= (VD'DV' + \lambda I)^{-1}VDU'y \\ &= V(D'D + \lambda I)^{-1}DU'y.\end{aligned}$$

Contrast this with the OLS estimator:

$$\begin{aligned}(X'X)^{-1}X'y &= (VD'DV')^{-1}VDU'y \\ &= V(D'D)^{-1}DU'y,\end{aligned}$$

which requires $D'D$ to be non-singular.

There is an interesting connection between Ridge regression and principal components (see Section 3.4 in the book by Hastie, Tibshirani and Friedman).

Choosing λ. As in Lasso, λ drives the bias/variance trade-off: small λ means high variance, high λ means high bias.

A popular approach is to select λ using cross-validation.

The limit of Ridge as λ tends to zero: "Ridgeless" regression

Least squares when $p > N$. When $p > N$ OLS is not uniquely defined. There are multiple solutions b to the system

$$y = Xb. \tag{1}$$

Let $\widehat{\beta}$ denote one solution. Then, if b is such that $b - \widehat{\beta}$ belongs to the null-space of A (that is, $A = (b - \widehat{\beta})$), then b is another solution.

Coding: write a computer code to compute the null-space of a matrix A, by making use of the SVD of A.

Ridgeless estimator. Let $\widehat{\beta}^*$ denote the solution to (1) that has minimum Euclidean norm. This is the "smallest" least squares estimator.

It turns out that $\widehat{\beta}^*$ coincides with the limit as λ tends to zero of the Ridge estimator $\widehat{\beta}^{Ridge}(\lambda)$ associated with the penalty λ. Hence the name "Ridgeless".

There is growing interest in this estimator, since it appears to have good out of sample prediction properties (this relates to the "double descent" phenomenon documented by Misha Belkin and others).

Variants of the Lasso

Group Lasso. Suppose we have now two linear equations:

$$y_{i1} = \sum_{j=1}^{p} \beta j1 x_{ij1} + u_{i1}$$
$$y_{i2} = \sum_{j=1}^{p} \beta j2 x_{ij2} + u_{i2}.$$

An example is a panel data model with 2 periods.

In some applications one wishes to impose that (β_{j1}, β_{j2}) is either jointly zero, or jointly non-zero.

The group lasso penalty is then:

$$\sum_{j=1}^{p} \sqrt{b_{j1}^2 + b_{j2}^2}.$$

Note that this is also:

$$\sum_{j=1}^{p} \|b_j\|_2.$$

The objective functions is still convex, and the properties are similar to the ones of the Lasso.

Fused Lasso. Consider a time-series model:

$$y_t = \beta_t + u_t,$$

where u_t is an iid shock, and β_t is very persistent.

For example, we might want to smooth the evolution of GDP or inflation.

Using the fused Lasso, one can impose that β_t changes only a few times during the observation sample, by minimizing:

$$\sum_{t=1}^{T}(y_t - b_t)^2 + \lambda \sum_{t=2}^{T} |b_t - b_{t-1}|.$$

Here we are penalizing the increments of b_t.

This idea can be generalized to allow for time-varying coefficients in a regression, say $\beta_t x_t$ for example.

Economic applications of the Lasso

Recovering network links from panel data

Panel data model. Read Manresa (2013):

$$y_{it} = w_{it}'\delta + \sum_{j=1}^{p} \gamma_{ij} x_{jt} + \alpha_i + u_{it}.$$

Here y_{it} may be some measure of productivity of firm i at time t, x_{jt} is the amount of R& D innovation that firm j does in period t, and w_{it} are other determinants of productivity. α_i is a firm-specific fixed effect.

Panel Lasso estimator. Manresa estimates this model by minimizing:

$$\sum_{i=1}^{N}\sum_{t=1}^{T}(y_{it} - w_{it}'\delta - \sum_{j=1}^{p} \gamma_{ij} x_{jt} - \alpha_i)^2 + \lambda \sum_{j \neq i} |\gamma_{ij}|.$$

For simplicity assume that δ is known.

We can interpret this estimator as a collection of N Lasso estimators, each of which minimizes:

$$\sum_{t=1}^{T}(y_{it} - w_{it}'\delta - \sum_{j=1}^{p} \gamma_{ij} x_{jt} - \alpha_i)^2 + \lambda \sum_{j \neq i} |\gamma_{ij}|.$$

The objective function is convex and can be minimized using cyclic coordinate descent.

Application: R&D spillovers. The panel Lasso estimator will enforce that, for each firm j, most of the $\widehat{\gamma}_{ij}$ are zero.

Hence, each firm is responsive to at most a few R& D investments from other firms.

Sparsity is plausible in this setting, since it is likely that forming a link with another firm (which would allow firm i to benefit from j's research) is costly.

Sparse bounded rationality

The sparse max operator. Read Gabaix (2014).

Is economic data sparse?

Read Giannone, Lenza and Primiceri (2017): "The Illusion of Sparsity".

Chapter 3: Factor methods and matrix completion

1. **PCA and factor models**

2. **Matrix Lasso**

3. **Economic applications of factor methods**

PCA and factor models

Principal components analysis

Principal components. Consider a (possibly random) matrix X, e.g. a matrix of regressors. The dimensions of X are $N \times p$.

From the SVD of X we have

$$X = UDV'.$$

Note that we can take U to be $N \times p$, D to be $p \times p$, and V to be $p \times p$. Let us order the diagonal elements of D (which are non-negative) as $d_1 \geq ... \geq d_p \geq 0$.

The columns of V are also the eigenvectors of $X'X = VD^2V'$. $v_1, ..., v_p$ are called the principal component directions of X.

The first eigenvector v_1 is such that Xv_1 has the largest variance among all (normalized) linear combinations of X. Note that $Xv_1 = d_1u_1$. u_1 is the first (normalized) principal component of X.

Likewise, u_2 is the second (normalized) principal component, and so on.

In many cases, the first few principal components explain most of the variation in X.

Principal components calculation. Suppose one wants to compute the first principal component of X. One possibility is to minimize:

$$\sum_{i=1}^{N}\sum_{j=1}^{p}(x_{ij} - \lambda_i f_j)^2,$$

subject to one normalization such as $(1/p)\sum_{j=1}^{p} f_j = 1$.

The solutions are $\lambda = d_1u_1$ (the first principal component, multiplied by the first singular value of X) and $f = v_1$ (the first principal component direction of X).

This expression can be generalized to other principal components.

Note that this least-squares representation may not be directly helpful for computation, since the objective function is not convex, and there are efficient algorithms to perform SVD.

However, the equivalence between PCA and the least squares problem is very useful to study interesting extensions of PCA.

Coding. verify that PCA is the solution of the above least squares problem.

Principal components regression. In regression models, a common way to reduce the dimensionality of X is to perform a PCA on its columns, and extract the principal components $u_1, u_2, ...$

Then, one regresses y on $u_1, ..., u_K$, where $K \leq p$. This is PC regression.

When $K = p$, PC regression is simply OLS. When $K < p$ this is a regularized version of OLS.

As in Lasso and series, K is a tuning parameter that governs the amount of regularization/dimension reduction.

Since the u_i are orthogonal, regressing y on $u_1, ..., u_K$ is equivalent to K univariate regressions on u_1, u_2,..., and u_K.

Static and dynamic factor models

Static factor models. We are now going to apply PCA methods and PCA-related methods in a context where one has N units followed over T periods.

For example, we may have $N = 20$ portfolios with prices varying over $T = 500$ days.

Alternatively, we may observed $N = 1000$ individuals for $T = 10$ periods, as in many panel data applications.

The model is then:

$$y_{it} = \mu_i' f_t + u_{it},\ i = 1, ..., N,\ t = 1, ..., T,$$

where μ_i and f_t are $K \times 1$ (note: both are column vectors).

K is the number of factors.

For example, when $K = 1$ we have:

$$y_{it} = \mu_i f_t + u_{it},$$

where μ_i and f_t are scalar. This is a one-factor model.

We are going to treat $\mu_1, ..., \mu_N$ and $f_1, ..., f_T$ as parameters ("fixed-effects").

A challenge is that the number of parameters increases with the sample size.

Matrix form. Let K be the number of factors. In matrix form, let us construct the matrices F (which is $T \times K$) and M which is $N \times K$), by concatenating the f_t's and the μ_i's. Let us also construct the matrices Y $(N \times T)$ and U $(N \times T)$, by concatenating the y_i's and the u_i's.

The factor model is then:

$$Y = MF' + U.$$

Normalization. At best we can consistently estimate MF'. If Ω is $K \times K$ orthogonal, then $MF' = M\Omega\Omega'F' = (M\Omega)(F\Omega)'$. Hence M and F are at best identified up to right-multiplication by an orthogonal matrix. Identification holds up to a choice of "rotation". A standard normalization is to impose $F'F = I_K$.

Static models: estimation.

$$\min_{\mu_1,...,\mu_N,f_1,...,f_T} \sum_{i=1}^{N}\sum_{t=1}^{T}(y_{it} - \mu_i' f_t)^2,$$

subject to a normalization, as indicated above.

The solution is PCA. In particular, the estimates of f_t are the principal component directions of Y. The estimates of μ_i are the principal components of Y, multiplied by the singular values of X.

Large factor models: properties. Bai and Ng (2002, Econometrica) study the properties of principal components estimates in a model where N and T tend to infinity simultaneously.

Intuitively, in this model it is necessary to have both large N (to learn about f_t) and large T (to learn about μ_i).

Using the normalization $F'F/T + I_k$, they find, for all factors $k = 1, ..., K$:

$$\sqrt{T}(\widehat{\mu}_{ik} - \mu_{ik}) \xrightarrow{d} \mathcal{N}(0, V_\mu),$$

and

$$\sqrt{N}(\widehat{f}_{tk} - f_{tk}) \xrightarrow{d} \mathcal{N}(0, V_f).$$

In these expressions, the true μ_i and f_t are suitably normalized. (recall the lack of identification in the absence of such normalization)

Moreover, the asymptotic convergence relies on some assumptions about the relative rates of growth of N and T. Bai and Ng's results hold when T and N are of the same order asymptotically, i.e. when T/N is constant in the limit.

Bai and Ng also propose a method to estimate the number of factors.

Weak dependence. The intuition behind Bai and Ng's theory is that $Y = MF' + U$, where MF' has rank K, and U are errors. Provided that u_{it} are sufficiently "weakly correlated" across both i and t, one can learn about M and F from Y.

Weak correlation over time could mean an MA process, or an AR process with autoregressive coefficient $|\rho| < 1$.

Weak correlation across units is harder to visualize. A possible dependence structure is obtained if one specifies a "spatial weights matrix" W (of dimensions $N \times 1$) such that w_{ij} represents the dependence of y_{it} on y_{jt}. We typically normalize the diagonal of W to have zero elements. A spatial AR(1) model is $y_t = Wy_t + \varepsilon_t$. Under suitable conditions of the singular values of W, such a process is weakly dependent across i.

Dynamic factor models. The static factor model makes no assumptions of μ_i or f_t.

This makes using for the model for prediction difficult: how to predict f_t in the future?

This limitation is particularly severe in macroeconomics, when using factor models to predict inflation, GDP, ...

In a dynamic factor model, we add a specification of the process $f_t \mid f_{t-1}, ...$ Typically, the process is specified as an AR(p) or Vector AR(p).

Estimation can be based on maximum likelihood. Estimators are typically consistent for fixed N as T tends to infinity under correct specifications.

Interactive fixed-effects model

Bai's model. Bai (2009) considers the model:

$$y_{it} = x_{it}'\beta + \mu_i' f_t + u_{it},$$

where μ_i, f_t are unrestricted.

This model has appeal when focusing on β, the "effect" of x on y, while attempting to control for a rich set of factors.

Bai's estimator is:

$$\min_{\beta, \mu_1, ..., \mu_N, f_1, ..., f_T} \sum_{i=1}^{N} \sum_{t=1}^{T} (y_{it} - x_{it}'\beta - \mu_i' f_t)^2,$$

subject to a normalization as above.

The rationale for this is estimator the above equivalence between PCA and the least-squares formulation. Here we simply add a covariate.

This estimation problem is not convex. Bai's strategy is to iterate between ordinary least squares for β (given μ and f) and PCA for μ, f (given β).

This iteration will reach a local minimum of the objective function. In practice, it is recommended to use several starting values.

Finally, we need an estimate of the number K of factors. This can be based on the Bai and Ng information criteria.

Bai's model: properties. As in Bai and Ng (2002), Bai considers an asymptotic sequence where N, T tends to infinity simultaneously.

He shows that:

$$\sqrt{NT}(\widehat{\beta} - \beta) \xrightarrow{d} \mathcal{N}(0, V_\beta),$$

where the expression for V_β is available in the paper.

This is a nice result, since it shows that $\widehat{\beta}$ has a standard distribution.

Pesaran's model. There are other models related to Bai's interactive fixed-effects model in the literature. In particular, Pesaran (2006) considers a model of the form:

$$y_{it} = x_{it}'\beta + \mu_i' f_t + u_{it}, \quad x_{it} = H_i f_t + v_{it}.$$

The difference with Bai's model is that he assumes that the factors f_t also enter the x_{it}'s. This simplifies estimation (but requires making an additional assumption).

The estimator Pesaran proposes, which is called "common correlated effects", is very simple to implement.

Matrix penalization methods

Nuclear norm regularization

Read Chapter 7 in Hastie, Tibshirani and Wainwright's book.

Nuclear norm of a matrix. The nuclear norm of a matrix is the analog of the ℓ^1 norm of a vector.

Specifically, let $M = UDV'$ be the SVD of M.

The nuclear norm is $\|M\|_1 = \operatorname{Tr} D$. This the sum of singular values of M.

Matrix Lasso estimator. Let Y be a matrix of data. For example, Y is $N \times T$ with entries y_{it}.

Consider the following penalized regression problem:

$$\min_{m_{11},...,m_{NT}} \|Y - M\|^2 + \lambda \|M\|_1.$$

This is a matrix analog of the Lasso. It is referred to as "nuclear norm regularization".

The matrix Lasso objective is convex. It turns out that the nuclear norm $\|M\|_1$ is a convex function of the elements of M. The proof is not obvious and we will not study it.

The convexity of the nuclear norm implies that the matrix Lasso objective function is convex. This is useful since we will not have to deal with local optima when attempting to compute the solution.

Computation. A simple way to compute the matrix Lasso estimator, for fixed λ, is to use an iterative algorithm inspired by the cyclic coordinate descent algorithm for Lasso. See Hastie, Tibshirani and Wainwright, page 175-176.

The algorithm highlights an intuitive link between nuclear norm regularization and thresholding of singular values, i.e. replacing d_k by $(d_k - \lambda)^+$.

In fact, in the case of:

$$\min_{m_{11},...,m_{NT}} \frac{1}{2}\|Y - M\|^2 + \lambda \|M\|_1,$$

we have a nice explicit result due to Cai, Candes and Shen (2008):

$$\widehat{M} = U D_\lambda V',$$

where $Y = UDV'$ is the SVD of Y, and the elements of the diagonal D_λ are $d_j(\lambda) = (d_j - \lambda)^+$.

$\widehat{M}$ is obtained by application of the "singular value shrinkage operator".

Coding. Implement matrix Lasso estimation.

Choice of λ. One may set λ by cross-validation. An alternative approach, as in Lasso, is to use a formula motivated by the asymptotic analysis.

Matrix completion

The Netflix movie challenge. Netflix launched a competition in 2006. The netflix dataset has $N = 17770$ movies and $T = 480189$ customers. The data contains ratings of movies. However the missing data problem is very severe: less than 1% of the ratings are present.

The goal is to predict the ratings for the unrated movies.

A possible objective is to minimize the rank of M (which is $N \times T$), subject to $m_{ij} = z_{ij}$ for the available entries $(i, j) \in \Omega$.

Such a "low-rank heuristic" provides good prediction results for the netflix challenge.

Matrix completion using nuclear norm. Minimizing the rank of a matrix subject to linear constraints is a very challenging problem.

A convex relaxation is to use the nuclear norm and solve:

$$\min_{M} \sum_{(i,j)\in\Omega} (z_{ij} - m_{ij})^2 + \lambda \|M\|_*.$$

This is a convex objective function.

Economic applications of factor methods

Macroeconomic forecasting

Stock and Watson's dynamic factor models. Sargent and Sims (1977) showed that two dynamic factors could explain a large fraction of the variance of important U.S. quarterly macroeconomic variables, including output, employment, and prices.

In a series of papers, Stock and Watson extended these approaches to jointly model a large number of macroeconomic time series, and use dynamic factor models for forecasting purposes. Stock and Watson (2011, Oxford handbook on economic forecasting) provides a survey of this line of work.

Stock and Watson (2012, NBER): a new factor in the Great recession. Stock and Watson (2012) analyze the macroeconomic dynamics of the 2007-09 recession in the US and the subsequent slow recovery. They use a dynamic factor model with 200 variables. They find that the Great recession was associated with the emergence of a new – financial – factor.

Generalizing differences-in-differences

Traditional Diff in Diff. Panel data differences-in-differences is a leading framework for empirical work in economics. The model is:

$$y_{it} = x_{it}'\beta + \mu_i + f_t + u_{it}.$$

The parameters μ_i, f_t are unrestricted. However, unlike in the interactive fixed-effects model, these parameters enter the model additively.

The key assumption in diif-in-diff is that, when $x_{it} = x$, $x'\beta + \mu_i + f_t$ is a common, parallel time pattern across individuals. This is referred to as the "common pre-trends assumption".

The standard estimator relies on double differencing: take out the mean over t and the mean over i, and add the grand mean.

Diff in Diff with factors. Using the methods in this chapter, we are able to relax the common pre-trends assumption.

Indeed consider Bai's model:

$$y_{it} = x_{it}'\beta + \mu_i' f_t + u_{it},$$

or Bai's model with additional additive parameters (with suitable normalizations):

$$y_{it} = x_{it}'\beta + \mu_i' f_t + a_i + b_t + u_{it}.$$

These models are easy to estimate using Bai's method. This can help relax the key assumption in diff-in-diff.

Another generalization of diff in diff: synthetic control. Read Adadie, Diamond and Hainmueller (2010).

Abadie's synthetic control method is closely related to factor models.

Indeed, in their JASA paper they use a one-factor model (with $K = 1$) to motivate their synthetic control estimator.

An alternative to synthetic control is to estimate a one-factor (or multiple-factor) model, as we have learned in this chapter.

Chapter 4: Nonlinear optimization

1. **Neural networks**
2. **Variational approximations**
3. **Economic applications**

Neural networks

Neural network models

Nonlinear regression: basis of functions. A popular approach to nonlinear regression is to consider a basis of functions, and consider the following model:

$$y_i = \sum_{k=0}^{K} a_k \phi_k(x_i) + u_i.$$

As we saw, example of functions ϕ_k are ordinary polynomials, orthogonal polynomials, splines, and wavelets.

Nonlinear regression: adaptive basis. Here we consider a modification of this approach, where we aim at estimating the basis function.

Formally, we consider a model of the form:

$$y_i = \sum_{k=0}^{K} a_k \phi_k(x_i; \theta_k) + u_i.$$

As an example, ϕ_k could be the density of a normal distribution, and θ_k could denote its mean and variance.

Other popular examples are neural networks, which we consider in this chapter, and trees, which we will focus on in the next chapter.

The sigmoid function. Neural networks make extensive use of link functions. A link function is a nonlinear function that is used to specify a basis.

A popular example is the sigmoid:

$$\sigma(v) = \frac{\exp(v)}{\exp(v)+1}.$$

Note that σ is simply the cdf of the standard logistic. It is increasing from 0 to 1.

Another, increasingly popular choice is the ReLU (rectified linear unit) function:

$$\sigma(v) = \max(0, v).$$

Unlike the sigmoid, ReLU is not differentiable. This is not a problem when one uses stochastic gradient descent for optimization (see below).

Single-layer neural networks. A simple neural network is as follows. We start with the vector x_i, which is $p \times 1$.

We form K linear combinations of them:

$$z_{ik} = \sum_{j=1}^{p} \gamma_{jk} x_{ij}, \quad k = 1, ..., K.$$

Finally, we combine the z_k as follows, and write:

$$y_i = \sum_{k=1}^{K} a_k \sigma(z_{ik}) + u_i.$$

Note: whether the sum ranges for 0 to K or from 1 to K is immaterial.

Multiple-layer neural networks. This idea can be generalized to multiple-layer neural networks. For example, the z_{ik}'s could be re-combined into new variable, say w_{im}, and so on...

This idea is at the core of deep learning methods.

A recent reference on deep learning is the book by Goodfellow, Bengio and Courville.

Properties

Approximation. The first property we know about neural networks is that they have good approximation properties. Any suitable function (continuous) can be approximated uniformly well with a single-layer neural network with a sufficient number of nodes. This result is due to Hal White, the inventor of the White formula in econometrics.

Statistical properties: parametric model. Neural networks are nonlinear regression models. The asymptotic theory of nonlinear regression is well understood.

In a model of the form:

$$y_i = m(x_i, \theta) + u_i,$$

with $\mathbb{E}(u_i \,|\, x_i) = 0$, let:

$$\widehat{\theta} = \underset{c}{\operatorname{argmin}} \sum_{i=1}^{N} (y_i - m(x_i, c))^2.$$

We have, under commonly used conditions:

$$\sqrt{N}(\widehat{\theta} - \theta) \xrightarrow{d} \mathcal{N}(0, V),$$

where V takes a "sandwich" form. In particular, its denominator involves the gradient of the function m with respect to θ.

Statistical properties: nonparametric model. At the same time, this theory abstracts from some of the key feature of neural networks.

-Neural networks are supposed to be "flexible" approximation. It is appealing to think of the true function $m(x)$ to be nonparametric.

Recent progress on this issue has been made by Farell, Liang, Misra (2020).

-Neural networks have many parameters in most applications. This causes problems such as high variance, multi-collinearity, overfit, ...

-Given the nonlinear nature of the objective function it is often unrealistic to think we can compute its global minimum. Yet the standard theory assumes that one has computed the global minimum.

Developing a statistical theory that accounts for these features is still an open problem.

Computation

Gradient descent. The goal is to compute the solution to a problem of the form:

$$\underset{c}{\operatorname{argmin}}\, Q(c) = \sum_{i=1}^{N} (y_i - m(x_i, c))^2.$$

A simple idea is to use Newton's method. Starting at c_0 we compute:

$$c_1 = c_0 - [Q''(c_0)]^{-1} Q'(c_0).$$

Then we proceed iteratively using:

$$c_{s+1} = c_s - [Q''(c_s)]^{-1} Q'(c_s),$$

until c_s and c_{s+1} are close enough relative to a given distance measure (for example: the maximum of the absolute value difference across components is small enough).

This method is called Newton-Raphson. It requires not only knowledge of the gradient Q' but also of the derivative Q''.

There are alternative methods that do not rely on Q'', e.g. methods based on line search.

Computing derivatives: backpropagation. A key challenge in implementing gradient descent in neural networks is to compute Q' (one generally finds a way not to have to compute Q'').

A trick is to use the chain rule of derivation in a smart way. This is called "backpropagation". Implementing this efficiently may reduce the computational cost substantially.

Local optima. Neural network objectives are typically non-convex. Gradient descent methods will then often converge to local optima only, not the global optimum.

A strategy is often to specify the initial values in a way that the response function is approximately linear. However there is so far no theory for why starting from this region is desirable.

Coding. Estimate the parameters of a CES production function.

Stochastic gradient descent

Computational cost in gradient descent? To assess the computational cost, it is useful to distinguish two challenging situations. In big data situations there are many observations (N is large). In big models situations there are many parameters (p is large).

When both N and p are large, as in neural network applications, it is useful to find ways to work with a smaller sample.

Batch methods. In a batch method we use an update rule that only depends on a subsample of the data. For example, we can implement Newton's step using a subsample to compute Q.

Robbins and Monro (1951) proposed to use different, randomly drawn subsamples in each iteration. This method is now called stochastic gradient descent.

The trade-off is an improved computational efficiency (much faster computations in big data situations), but the method is stochastic and depends on a key tuning parameter: the "learning rate".

Formally, we compute:

$$c_{s+1} = c_s - \mu_s Q'_s(c_s),$$

where Q_s is based on a subsample of the data.

The learning rate μ_s is such that it should tend to zero as s tends to infinity, but not too fast. If it tends too fast then we do not learn from all of our data. If it tends too slowly to zero then the method will converge very slowly and the computational advantages will be lost.

This trade-off has been studied theoretically, and default choices exist. However stochastic gradient descent methods are often sensitive to this choice.

Stochastic gradient descent is widely used to estimate ("train") neural networks.

Variational approximations

Intractable objectives: examples. In many case it is complicated to compute the function Q.

For example, Q may be the likelihood function of an economic model. Often the likelihood function depends on unobservables (shocks, individual heterogeneity...) and computing it requires multiple integrations that may not be feasible.

As another example, popular in machine learning and statistics, drawing from a joint posterior distribution in many dimensions may be numerically challenging.

Variational inference. VI is based on finding a lower bound on the objective function, and maximizing this lower bound.

The lower bound exploits properties of the Kullback-Leibler divergence.

A typical approach is to work with a family of objective functions that can be factored in parameters, such as:

$$Q(c_1, c_2) \approx Q_1(c_1)Q_2(c_2).$$

This is only an approximation. On the other hand, it may be much easier to optimize $Q_1(c_1)Q_2(c_2)$ since that can be done separately for c_1 and c_2.

What do we recover? The variational estimator is not consistent for the true parameter value in general. This is because we are optimizing the "wrong" objective function.

This still may perform fine in machine learning since the goal of the analysis (out of sample prediction) is very clear and can be checked. However, this inconsistency is

problematic in economics.

Economic applications

Network models of link formation

Network formation. Active literature. Friendship, links between firms, banks... Logit models (Graham). Models with network externalities (Mele).

Stochastic blockmodel.

$$D_{ij} = \mathbf{1}\{U(X_{ij}, A_i, A_j, \varepsilon_{ij}) > 0\},$$

where ε_{ij} are i.i.d. across (i, j) pairs. More general than fixed-effects models.

Maximum likelihood? Intractable likelihood objective.

Variational approach. Postulate a model that is independent across pairs. Mean field approximation. Optimization. There are statistical guarantees in this context (Bickel and co-authors, 2013).

Text analysis

Text data. Increasingly popular. Read Gentzkow and Shapiro, "Text as Data".

Latent Dirichlet allocation. A document is a mixture of topics. Goal: recover topic distributions. Example: Blei and Lafferty's analysis of *Science* magazine.

The LDA model. Hierarchical model.

Maximum likelihood? Intractable likelihood objective.

Variational approach. Mean field approximation. Optimization. In this case statistical guarantees are not obvious.

Chapter 5: Partitioning methods

1. **Decision trees**
2. **K-means clustering**
3. **Economic applications of partitioning methods**

Decision trees

Partitioning covariates

Nonlinear regression. We focus on a nonlinear regression model:

$$y_i = f(x_i) + u_i.$$

Unlike basis expansion methods, here we allow for models where the choice of basis is adaptive, i.e. where we estimate the basis.

Step functions. A simple possibility is to use a step function estimator. In the univariate case this amounts to partitioning the real line or a compact interval. In the multivariate case there is a curse of dimensionality, since cells may be empty.

Optimal steps? When choosing the number of cells we face the standard bias-variance trade-off. Cross-validation is an option to select the step size.

Nadaraya Watson and kernel methods. A related possibility is Nadaraya-Watson. With a kernel function κ and a bandwidth h we compute:

$$\widehat{f}(x) = \frac{\sum_{i=1}^N \kappa(\frac{x-x_i}{h}) y_i}{\sum_{i=1}^N \kappa(\frac{x-x_i}{h})}.$$

Properties have been studied extensively. The bias is $O(h^{-2})$ for standard kernels, the variance is $O(1/(Nh))$.

An issue with kernel is that the bandwidth does not depend on x. It is possible to allow for "adaptive bandwidths", but theory and implementation is harder.

Nearest neighbors. A popular adaptive method is K-nearest neighbors.

$$\widehat{f}(x) = \frac{\sum_{i=1}^{N} \mathbf{1}\{x_i \in B_x\} y_i}{\sum_{i=1}^{N} \mathbf{1}\{x_i \in B_x\}},$$

where B_x consists of the K points x_i closest to x according to some metric (example: in terms of Euclidean norm).

This is adaptive, since the width of B_x varies with how dense the data is around x.

For consistency we need two key conditions: K tends to infinity, and diameter(B_x) tends to zero.

CART

Objective function. CART (classification and regression trees) was proposed by Breiman et al in their classic book (1984). The idea of a tree is to minimize:

$$\sum_{i=1}^{N} \left\| y_i - \widetilde{h}(k_i) \right\|^2,$$

subject to the k_i forming a partition of the X space that is not "too complex".

The restriction will be that the partition is composed of rectangles in the X space.

Greedy algorithm. In CART we first estimate a "deep" tree. The idea is as follows:

-Go though all possible covariates,

-and go through all possible splits of the covariates. A split is a threshold c that gives rise to two sub-samples $x_{ik} \leq c$ and $x_{ik} > c$.

-Choose the covariate and split that gives the lowest value of the within-group sum-of-squares.

-Continue until each leaf of the tree has one element.

Trees have a recursive structure that is helpful for fast computation, and interpretable (example: used for medical decision making).

Note that this algorithm is greedy, i.e. non-optimal. Finding the optimal partition among all sets of rectangles is a combinatorially hard problem.

Deep versus shallow trees. A deep tree is likely to overfit. At the bottom of the tree, all leafs have one observation so there is massive overfit.

In contrast, a shallow tree may have large leaves but the bias may be large. This is the familiar trade-off.

Pruning and cross-validation. In CART we first estimate a deep tree and then go back to prune some of the leaves of the tree. The idea is to consider all potential pruning of the deep tree as a set of potential trees, and to select the tree that achieves a desired fit/complexity balance.

A key result in Breiman's book shows that exploring the family of trees is in fact computationally feasible. Typically the chosen tree is based on a penalty term that penalizes the number of leaves. That term is often chosen by cross-validation.

The whole procedure, including the CV step, is typically very fast.

Performance of tree methods

A difficulty. Trees may be inefficient is some standard models. Consider for example an additive model of the form: $\mathbb{E}(y_i \,|\, x_{i1}, x_{i2}) = x_{i1} + x_{i2}$. In this case one will typically need lots of rectangles to approximate the conditional expectation well. An obvious solution would be to add $x_{i1} + x_{i2}$ as a covariate, but it may be hard to spot whether a particular interaction is suitable or not. Neural networks address this concern to some extent by working with more flexible bases.

Coding. Estimate a trees on an additive DGP.

Statistical guarantees. The literature provides some results on consistency, and convergence rates. However, convergence rates are the same as other methods if one does not restrict the DGP.

Honest splitting. A variation of standard tree methods is to use "honest" splits. This consists in using a subsample to form the partition, and another subsample (typically, the complement of the first one in the whole sample) to estimate the mean outcomes within each group.

K-means clustering

K-means estimator and algorithms

To obtain a partition with K elements, the k-means algorithm is useful.

The k-means algorithm is obtained by:

$$\left(\widehat{h}, \widehat{k}_1, ..., \widehat{k}_N\right) = \underset{\left(\widetilde{h}, k_1, ..., k_N\right)}{\operatorname{argmin}} \sum_{i=1}^{N} \left\| x_i - \widetilde{h}(k_i) \right\|^2.$$

The minimum is over all possible partitions of units in at most K groups. Computing a global minimum may be challenging, yet fast and stable heuristic algorithms have been developed, such as iterative descent, genetic algorithms or variable neighborhood search. Lloyd's algorithm is often considered to be a simple and reliable benchmark (see Steinley, 2006, and Bonhomme and Manresa, 2015, for algorithms and references).

The result of k-means is a Voronoi Tesselation.

Lloyd's algorithm. Lloyd's algorithm iterates between two steps, until convergence:

1. Given $\widetilde{h}(1)$, ..., $\widetilde{h}(K)$, update k_i for all i.
2. Given $k_1, ..., k_N$, update $\widetilde{h}(1)$, ..., $\widetilde{h}(K)$ (those are simply the within-group means).

The algorithm is typically very fast. However, it depends on the initial parameter values. In practice, it is customary to start with 1000 or 10000 randomly generated vectors $\widetilde{h}(1)$, ..., $\widetilde{h}(K)$, and select the solution that yields the smallest value for the objective function.

What does K-means estimate?

Grouped heterogeneity. k-means clustering can perform well in DGPs with grouped heterogeneity. For example:

$$x_i = \mu_{k_i^0} + u_i,$$

where u_i i iid independent of $k_1^0, ..., k_N^0$. Those are the "true groups" in the population.

Coding. Estimate the k-means partition on a grouped DGP.

How to recover K? Two possibilities are often used: based on information criteria (such as BIC) or cross-validation.

Economic applications of partitioning methods

Exploring heterogeneity: paths of democracy.

The application. Read Bonhomme and Manresa (2015). The model is:

$$y_{it} = \theta x_{it} + \alpha_{k_i^0,t} + u_{it},$$

where y_{it} is an indicator of the quality of democracy in country i at time t, x_{it} is income, and $\alpha_{k_i^0,t}$ vary across groups and across time.

Bonhomme and Manresa identify 4 distinct groups of countries: stable democracies, stable autocracies, and two groups of countries that transitioned between autocracy and democracy at different times.

Inference on parameters. The existing theory (Bonhomme and Manresa, 2015) establishes consistency and asymptotic normality as N and T tends to infinity.

Under suitable conditions, the kmeans estimates of k_i^0 converge to k_i^0 in this asymptotic.

As a result, one can treat the estimated groups as if they were known.

More recent theory relaxes these assumptions, and allows one to study the properties of k-means in settings where heterogeneity is not discrete (Bonhomme, Lamadon and Manresa, 2017).

Estimating heterogeneous treatment effects

A first application is to document heterogeneity in treatment effects under a selection on observables assumptions, see Athey and Imbens (2016).

An important modification of the standard tree for these applications is "honest splitting". This allows one to compute asymptotically valid confidence intervals.

Chapter 6: Re-sampling methods

1. **The bootstrap**

2. **Bagging and random forests**

The bootstrap

The bootstrap principle

Monte Carlo simulations. Draw S samples at random from the DGP. Estimate $\widehat{\theta}^s$ in each sample.

Bootstrapping an estimator. Draw S samples at random uniformly from the original sample, with replacement. Estimate $\widehat{\theta}^s$ in each sample.

The percentile method. report the 2.5 and 97.5 percentiles of $\widehat{\theta}^s$ across s. Under general conditions this provides an asymptotically valid confidence interval for the true θ.

Coding. Implement the bootstrap in a linear regression with heteroskedasticity.

Variants of the bootstrap

Nonparametric bootstrap and sub-sampling. The basic bootstrap method is called the nonparametric bootstrap. It is due to Brad Efron.

An alternative is to draw subsamples of size $n < N$, without replacement, from the original sample. This is called subsampling.

Residual bootstrap, wild bootstrap and parametric bootstrap. There are other bootstrap methods that exploit additional structure on the model. A popular example is the wild bootstrap. In a regression model this amounts to multiplying the regression residuals $\widehat{u}_i$ by a normal(0,1) random variable η_i^s. And reporting regression estimates based on the new "data":

$$y_i^s = x_i'\widehat{\theta} + \eta_i^s \widehat{u}_i.$$

Bagging and random forests

Bagging estimators

The bagging principle. Most often, we use the bootstrap to compute standard errors and compute confidence intervals.

In bagging, the idea is to use the bootstrap for estimation.

Given a bootstrap sample y_i^s, x_i^s, $i = 1, ..., N$, we estimate a bootstrap estimator $\widehat{\theta}^{(s)}$.

Then we report $\frac{1}{S}\sum_{s=1}^{S}\widehat{\theta}^{(s)}$.

Bagging in linear models. Bagging does not help in linear models. An exception where bagging may be beneficial is to "smooth out" the impact of outlying observations.

Bagging in nonlinear models. Bagging may help in nonlinear models.

The insight is that bagging may reduce variance, by averaging.

For the variance reduction to be substantial, it is important that the bootstrap samples be not too correlated.

Ensuring weak correlation between bootstrap samples is the key idea behind random forests.

Bagging trees: random forests

Bagged trees. In a forest, we try to average the results of trees that are weakly correlated.

Usually, we combine two ideas to ensure weak correlation, both due to Breiman.

(1) We construct the trees on bootstrap samples of the original sample. Of course theses samples are not independent since they are based on the same underlying data.

(2) When going down the tree, when deciding which covariate to split, we select a set of M potential covariates along which the split can occur. This selection is at random. Then we proceed with the split as usual. We proceed in this way before every split.

(2) is the crucial step. Intuitively, the lower M the less correlated the trees are. Hence M is a key tuning parameter. When M is equal to the number of covariates, the result of bagging is the original deep tree.

Athey and Wager (2017) provide asymptotic theory for random forests. In the next chapter we will see some theory for partitioning methods.

Practical issues for random forests. In practice researchers often do not prune, and simply average the resulting deep trees.

The choice of M is key. A common rule is the number of covariates divided by three.

An example: estimating heterogeneous treatment effects. Read Athey and Wager (2017).