Parcours IA - 2019/2020

Fondements de l'Intelligence Artificielle et du *Machine Learning*

Nicolas Vayatis

Novembre 2019

Session 1 - Concepts fondamentaux

Supervised Machine Learning

Learning and information

The bias-variance trade-off

Empirical Risk Minimization

Penalized Empirical Risk Minimization

Supervised Machine Learning

Learning and information

Learning like the twenty-question game

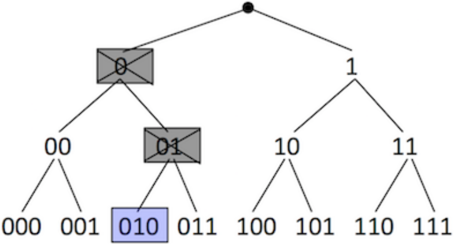
- Assume Nature has picked one function among K and we want to reveal this function
- Assume we have an oracle answering YES or NO when we ask a question about this function
- What is the optimal number n of questions to ask to find the unknown function?

Brute force learning
Finite case

- ISSUE: How many questions with answers YES or NO one has to ask the oracle to find THE function among K functions?
- STRATEGY: Proceed recursively by splitting the set of functions in two groups and asking whether THE function is the first group and removing the group which does not contain the function. This leads to the identification of the desired function with about $\log K$ questions.
- ANSWER: Number of questions $n = \dfrac{\log K}{\log 2}$
- NB: this quantity represents the number of bits of information characterizing the function in the set of K functions

Shannon's Information theory
The origin of the log K

- Related to the entropy of a distribution P in information theory: $H(P) = -\sum_{k=1}^{K} P(k) \log P(k)$
- The entropy is the number of bits to encode a collection of K symbols (functions)



From questions to data
Zero error case

- Notations: Domain space $\mathcal{X}$ and label space $\mathcal{Y} = \{0, 1\}$
- ISSUE: How many examples $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ are required to find among a *finite* collection (size K) of indicator functions $f : \mathcal{X} \rightarrow \{0, 1\}$ the desired one?
- SAME ANSWER: Number of examples $n = \dfrac{\log K}{\log 2}$
- STRATEGY: One has to find a vector x_i such that half of the functions take value 1 and the other half take value 0 and ask the oracle whether the desired function takes value 1 or 0 on this vector and discard those functions taking the opposite value. Apply this n times.

Probably approximately correct learning
Zero error case

- REMARK: it may be hard to find such an x_i which splits the collection of functions in two.
- NEW MODEL: Assume $X_1, \ldots, X_n$ is an IID sample
- QUESTION: How many examples (X_i, Y_i) are required to find among a finite collection of indicator functions $f : \mathcal{X} \to \{0, 1\}$ the one that with probability $1 - \delta$ is ε-close to the desired one?
- ANSWER: Number of examples

$$n = \frac{\log K - \log \delta}{\varepsilon}$$

(Proof left as an exercise)

Probably approximately correct learning
General case

- ASSUME: among K functions, NONE of them commits zero error on the sample (X_i, Y_i).
- SAME ISSUE AS BEFORE
- ANSWER: Number of examples on average

$$n = \frac{\log K - \log \delta}{\varepsilon^2}$$

Same dependency on K, the only change is in the constant.

(Proof coming next)

Questions raised

- Proof arguments for PAC learnability (finite case)
- PAC: From finite to infinite collection of functions
- From "strategies" to "learning algorithms"
- What is lost through sampling?

Supervised Machine Learning

The bias-variance decomposition in Machine Learning

Supervised learning setup

- Goal of learning: an *optimal* decision function $h^* : \mathcal{X} \to \mathcal{Y}$
  $\mathcal{X}$: domain set, $\mathcal{Y}$: label set

- Input of learning:
  - **Training data:** a set of labeled data

    $$D_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$$

    of size n, where the (X, Y)'s are in $\mathcal{X} \times \mathcal{Y}$
  - **Hypothesis space:** a collection $\mathcal{H}$ of candidate decision functions $h \ : \ \mathcal{X} \to \mathcal{Y}$

- Output of learning: an empirical decision function $\widehat{h}$ in the hypothesis space $\mathcal{H}$ estimated from training data D_n

Evaluating performance/error

- Loss function: $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, +\infty]$

  Examples :

  - classification loss $\ell(y, y') = \mathbb{I}\{y \neq y'\}$
  - square loss $\ell(y, y') = (y - y')^2$

- Assume (X, Y) random pair with distribution P

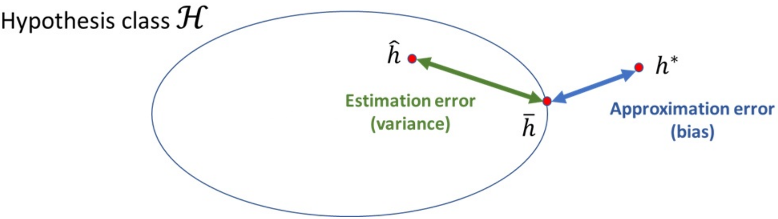
- True error of a decision rule h: this is a distribution-dependent functional

$$L(h) = \mathbb{E}\big(\ell(h(X), Y)\big) = \int \ell(h(x), y) dP(x, y)$$

The key trade-off in Machine Learning

- Denote by $L(h)$ the error measure for any decision function h
- Optimal rules: $L(\bar{h}) = \inf_{\mathcal{H}} L$, and $L(h^*) = \inf L$
- Bias-Variance type decomposition of error for any output $\widehat{h}$:

$$L(\widehat{h}) - L(h^*) = \underbrace{L(\widehat{h}) - L(\bar{h})}_{\text{estimation (stochastic)}} + \underbrace{L(\bar{h}) - L(h^*)}_{\text{approximation (deterministic)}}$$



About approximation error

- Cybenko (1989) - Denseness result in the spirit of Stone-Weierstrass showing that any linear combination of compositions of sigmoid with linear functions is dense wrt the supremum norm in the space of continuous functions over the *d*-dimensional unit cube.
- Barron (1994) - Approximation error bound involves a parameter quantifying the smoothness of the target function.
- Status of this question in the regression setup:
  - For kernel machines: a full theory is available thanks to Smale (2003), Steinwart (2008).
  - For deep learning: recent work by Grohs, Perekrestenko, Elbrächter, and Bölcskei (2019) .
  - In the classification setup, tough problem, still open issue...

The case of linear models with Gaussian noise

- Linear model in dimension d with vector notations (sample size n):
$$\mathbf{Y} = \mathbf{X}\beta^* + \varepsilon \in \mathbb{R}^n$$
where $\mathbf{X}$ of size $n \times d$, and ε is a gaussian vector mean zero, variance σ^2
- Least square estimator denoted by $\widehat{\beta}_{\mathbf{n}}$
- error of the LSE (expectation wrt the sample distribution)

$$\frac{1}{n}\mathbb{E}\big(\|\mathbf{X}\beta^* - \mathbf{X}\widehat{\beta}_{\mathbf{n}}\|^2\big) = \mathrm{Bias} + \sigma^2 \frac{d}{n}$$

Supervised Machine Learning

Empirical Risk Minimization (ERM)

The ERM principle
Definition

- Loss function: $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, +\infty]$
- Empirical risk of a decision rule h: this is a data-dependent functional

$$\widehat{L}_n(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(X_i), Y_i)$$

- ERM = Empirical Risk Minimization

  Learning from training data amounts to solving the following optimization problem

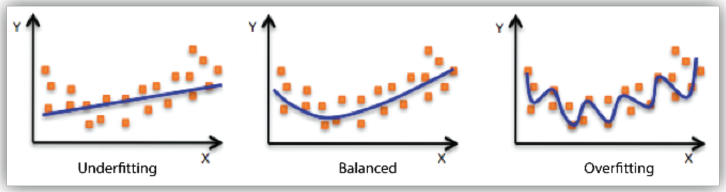
$$\widehat{h}_n = \arg\min_{h \in \mathcal{H}} \widehat{L}_n(h)$$

where the minimization is restricted to the hypothesis space.

The ERM principle
Main questions

1. The question of consistency: convergence of $\widehat{h}_n$ wrt the sample size n?
2. What is the cost incurred for using training data instead of the actual data?
3. What is the nature of the trade-off to calibrate the complexity of the hypothesis space $\mathcal{H}$?

Overfitting vs. underfitting



Less is more:

- It turns out that considering all measurable functions leads to overfitting $\Rightarrow \mathcal{H}$ has to be a restricted class!

But greed is good:

- Algorithms which have the capacity to overfit means they have high representation power (arbitrary small approximation error)

The notion of *true* error

- Assumption:

  (X, Y) is a pair of random variables with joint distribution P

- True error of a decision rule h: this is a distribution-dependent functional

$$L(h) = \mathbb{E}(\ell(h(X), Y)) = \int \ell(h(x), y) dP(x, y)$$

Optimal elements, consistency and bounds

- Bayes rule h^* and Bayes error L^*

$$h^* = \arg\min_h L(h) \quad \text{and} \quad L^* = L(h^*)$$

- (Strong) Consistency of an inference principle $\widehat{h}_n$

$$L(\widehat{h}_n) \rightarrow L^* \ , \quad \text{almost surely}$$

- The nonasymptotic bounds Eldorado:

$$L(\widehat{h}_n) - L^* \leq U(n, \mathcal{H}) \quad \text{whp}$$

Estimation vs. approximation error
Extension of bias-variance decomposition

- Proof idea: Add and retrieve $\widehat{L}(\widehat{h}_n)$, $\widehat{L}(\bar{h})$, $L(\bar{h})$, then use the definition of ERM to upper bound the sum. Difference between L and $\widehat{L}$ appear twice.
- We have:

$$L(\widehat{h}_n) - L^* \leq \underbrace{2 \sup_{h \in \mathcal{H}} |L(h) - \widehat{L}_n(h)|}_{\textit{estimation (stochastic)}} + \underbrace{L(\bar{h}) - L^*}_{\textit{approximation (deterministic)}}$$

Finite hypothesis class
Generalization error bound for ERM

- Assume that the hypothesis class $\mathcal{H}$ of decision functions is finite and $h^* \notin \mathcal{H}$
- Then, we have, for any δ, with probability at least $1 - \delta$:

$$\forall h \in \mathcal{H} \ , \quad L(h) \leq \widehat{L}_n(h) + \sqrt{\frac{\log |\mathcal{H}| + \log \left(\frac{1}{\delta}\right)}{2n}}$$

- $\log |\mathcal{H}| = \log K \rightarrow$ Statement in the introduction, see!

Finite hypothesis class
Sketch of proof

- Hoeffding's inequality:
  - Consider $Z_1, \dots, Z_n$ IID over $[0, 1]$ and $\overline{Z}_n = \frac{1}{n} \sum_{i=1}^{n} Z_i$
  - We have, for any $t > 0$

$$\mathbb{P}\{\overline{Z}_n - \mathbb{E}(Z_1) > \varepsilon\} \leq \exp(-2n\varepsilon^2)$$

- Union bound: For any two measurable sets A, B, we have:

$$\mathbb{P}\{A \cup B\} \leq \mathbb{P}\{A\} + \mathbb{P}\{B\}$$

Empirical risk minimization (ERM)

- Set the ERM classifier as:

$$\widehat{g}_n = \arg\min_{g\in\mathcal{G}} \widehat{L}_n(g)$$

- Define the best classifier in the class as:

$$\overline{g} = \arg\min_{g\in\mathcal{G}} L(g)$$

- We have:

$$L(\widehat{g}_n) - L(\overline{g}) \leq 2 \sup_{g\in\mathcal{G}} |L(g) - \widehat{L}_n(g)|$$

$\Rightarrow$ Need for uniform rates of convergence in the law of large numbers

The role of regularization

Penalized ERM

- Consider $\mathcal{G}_1, \mathcal{G}_2, \ldots$ a sequence of function classes
- Consider $\widehat{g}_1, \widehat{g}_2, \ldots$ the corresponding sequence of ERM
- Denote by $L_k^* = L(g_k^*) = \inf_{g \in \mathcal{G}_k} L(g)$ the optimal error in $\mathcal{G}_k$
- Penalized ERM estimator:

$$\widehat{g} = \widehat{g}_{\widehat{k}}$$

where

$$\widehat{k} = \operatorname*{arg\,min}_{k \geq 1} (\widehat{L}_n(\widehat{g}_k) + \widehat{C}_k)$$

and $\widehat{C}_1, \widehat{C}_2, \ldots$ a sequence of random (or fixed) *penalties*

The key to model selection

Theorem. (Model selection with random penalties (Lugosi and Wegkamp, 2004))

Suppose that random variables $\widehat{C}_1, \widehat{C}_2, \dots$ are such that:

$$\mathbb{P}(\widehat{C}_k \leq (L - \widehat{L}_n)(\widehat{g}_k)) \leq \frac{\gamma}{n^2 k^2}$$

and

$$\mathbb{P}(\widehat{C}_k \leq (\widehat{L}_n - L)(g_k^*)) \leq \frac{\gamma}{n^2 k^2}$$

for some $\gamma > 0$ and for all $k \geq 1$, $n \geq 1$.

Then, with probability at least $\left(1 - \frac{4\gamma}{n^2}\right)$:

$$L(\widehat{g}_n) - L^* \leq \inf_{k \geq 1} (L_k^* - L^* + 2\widehat{C}_k)$$

Sparsity and linear models

Tuning the dimension of the model

Linear regression model
Notations

- Vector notations:

  Response vector $\mathbf{Y} \in \mathbb{R}^n$, input data matrix $\mathbf{X}$ (size $n \times d$)

- Linear model with vector notations:

$$\mathbf{Y} = \mathbf{X}\beta^* + \varepsilon$$

  where ε random noise vector (centered, independent of $\mathbf{X}$)

The sparse linear regression model

- Intuition: what if there are uninformative variables in the model but we do not know which they are?
- Sparsity assumption: Let β^* the true parameter which only a subset of variables (called *support*)

$$m^* = \{j : \beta_j^* \neq 0\} \subset \{1, \dots, d\}$$

- ℓ_0 norm of any β: $\|\beta\|_0 = \sum_{j=1}^{d} \mathbb{I}\{\beta_i \neq 0\}$

Two possible formulations
Constrained vs. Penalized optimization

1. Ivanov formulation: take k between 0 and $\min\{n, d\}$

$$\min_{\beta \in \mathbb{R}^d} \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 \quad \text{subject to } \|\beta\|_0 \leq k$$

2. Tikhonov formulation: take $\lambda > 0$

$$\min_{\beta \in \mathbb{R}^d} \left\{ \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda \|\beta\|_0 \right\}$$

Comments

- Tikhonov looks as a Lagrange formulation of Ivanov
- But here the two formulations are NOT equivalent due to the lack of smoothness of the ℓ_0 norm
- Ivanov with ℓ_0 constraint is known as the Best Subset Selection problem for which there are algorithms based on heuristics (e.g. Forward Stagewise Regression) which work ok up to $k \simeq 35$. Recent advances: check Mixed Integer Optimization (MIO) formulation by Bertsimas et al. (2016).
- Focus here on Tikhonov regularization

Model selection in linear models

- Model: $\mathbf{Y} = \mathbf{X}\beta^* + \varepsilon$
- Consider a model for β^* that is a subset m of indices of $\{1, \dots, d\}$
- Example: In dimension $d = 3$, we have:
  - 1 model of size $|m| = 0$: constant model
  - 3 models of size $|m| = 1$: $\{1\}, \{2\}, \{3\}$
  - 3 models of size $|m| = 2$: $\{1, 2\}, \{2, 3\}, \{1, 3\}$
  - 1 model of size $|m| = 3$: $\{1, 2, 3\}$

  We potentially have 8 versions of Least Square Estimator (LSE), we call call constrained LSE (except for the case $|m| = 3$ which is unconstrained).

Model selection in linear models

- Model: $\mathbf{Y} = \mathbf{X}\beta^* + \varepsilon$

- Consider the set $\mathcal{M}$ of subsets m of the variables among indices $\{1, \ldots, d\}$. There are 2^d such sets m.

- For every $m \in \mathcal{M}$, there is a standard linear regression model with dimension $k_m = |m|$. In other words, for those $j \notin m$, we have $\beta_j^* = 0$.

- For each model $m \in \mathcal{M}$, compute the constrained Least Square Estimator $\widehat{\beta}_m$.

- The final estimator is the "best" among $\widehat{\beta}_m$ over all $m \in \mathcal{M}$

What "Best" actually means
The oracle

- Error given by: $r_m = \frac{1}{n}\mathbb{E}\big(\|\mathbf{X}\beta^* - \mathbf{X}\widehat{\beta}_m\|^2\big)$
- Best theoretical estimator (called *oracle*):

$$\widehat{\beta}_{\overline{m}} \quad \text{where} \quad \overline{m} = \underset{m \in \mathcal{M}}{\arg\min}\, r_m$$

Solving the computation burden
The power of convexity

- Practical methods for model selection are essentially greedy heuristics consisting in adding and/or retrieving one variable at the time to explore part of the whole model space which is exponential in the dimension. Examples are: Forward Stagewise Regression, Forward-Backward algorithm...

- Question: would it be possible to solve the optimization wrt the unknown parameter β AND wrt to its support subset of indices jointly?

- Answer is yes at the cost of the so-called relaxation of the non-convex formulation with the ℓ_0 penalty to a convexified problem with an ℓ_1 penalty.

The LASSO for linear models
From ℓ_0 to ℓ_1

- Consider the relaxation of the previous problem replacing the ℓ_0-norm by the ℓ_1-norm:

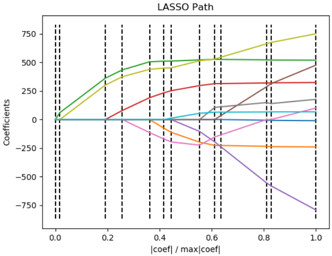
$$\|\beta\|_1 = \sum_{j=1}^{d} |\beta_j|$$

- The new estimator is called the LASSO: for any $\lambda > 0$,

$$\widehat{\beta}_\lambda \in \underset{\beta \in \mathbb{R}^d}{\arg\min} \left\{ \|\mathbf{Y} - \mathbf{X}\beta\|^2 + \lambda \|\beta\|_1 \right\}$$

Blessings of the LASSO

- Approximate solutions via efficient algorithms building the so-called regularization paths $\lambda \to \widehat{\beta}_\lambda$:



- Theoretical soundness: it can be shown that: as $n, d \to \infty$

$$\frac{1}{n}\mathbb{E}\left(\|\mathbf{X}\beta^* - \mathbf{X}\widehat{\beta}\|^2\right) \leq C\|\beta^*\|_1\sqrt{\frac{\log d}{n}}$$

The ”mother” of ML algorithms

Penalized optimization

- Learning process as the optimization of a data-dependent criterion:

$$\text{Criterion}(h) = \text{Training error}(h) + \lambda\ \text{Penalty}(h)$$

- Training error: data-fitting term related to a loss function
- Penalty: complexity of the decision function or function norms (e.g. LASSO)
- Constant λ: smoothing parameter tuned through cross-validation procedure

How to create ML algorithms?

- Standard function classes (e.g. linear functions) and risk (e.g. least squares) and variations on the penalties

$$\mathrm{Criterion}(h) = \mathrm{Training\ error}(h) + \lambda\ \mathrm{Penalty}(h)$$

(e.g. in least square minimization: LASSO, Group LASSO, Elastic Net, Fused LASSO, structures penalties...)

- Playing with losses changing the training error

$$\mathrm{Criterion}(h) = \mathrm{Training\ error}(h) + \lambda\ \mathrm{Penalty}(h)$$

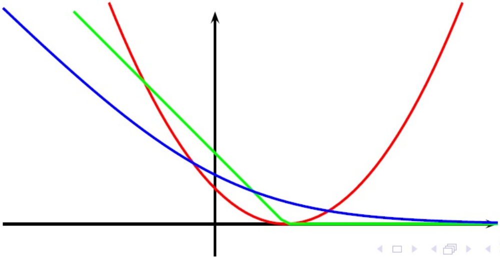
Changing loss functions

A few examples:

Ridge regression: $$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2}(y_i - \beta^\top \mathbf{x}_i)^2 + \lambda \|\beta\|_2^2.$$

Linear SVM: $$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i \beta^\top \mathbf{x}_i) + \lambda \|\beta\|_2^2.$$

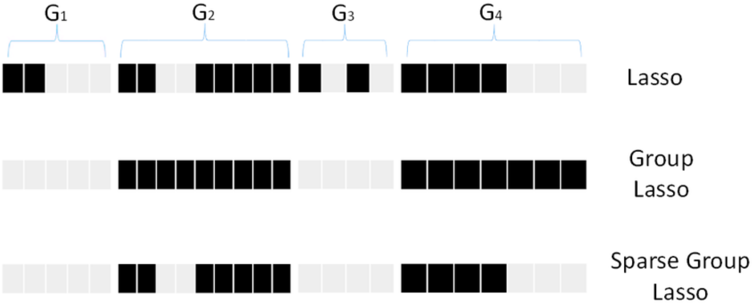
Logistic regression: $$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y_i \beta^\top \mathbf{x}_i}\right) + \lambda \|\beta\|_2^2.$$



Changing penalty

Structured sparsity in the case of linear models

Putting human priors in penalties
Sparsity patterns



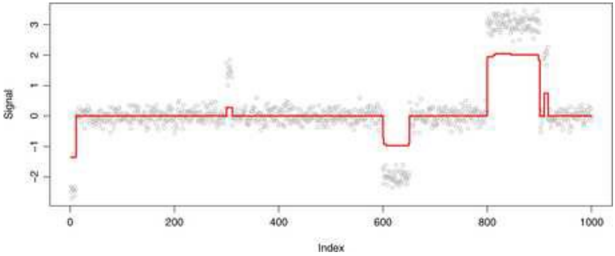
The simplest structured penalty
Group LASSO

- Group structure on the parameter β^*: let G the number of groups of subsets of indices in $\{1, \dots, d\}$ and, for $g = 1, \dots, G$, we denote by $\mathbf{X}^{(g)}$ the submatrix of $\mathbf{X}$ with variables in group g and by $\beta^{(g)}$ the coefficient vector applied to variables in group g and d_g is the size of group g.

- Group LASSO formulation:

$$\widehat{\beta}_\lambda \in \operatorname*{arg\,min}_{\beta \in \mathbb{R}^d} \left\{ \|\mathbf{Y} - \mathbf{X}\beta\|^2 + \lambda \sum_{g=1}^{G} \sqrt{d_g} \|\beta^{(g)}\| \right\}$$

Case of temporal patterns
Fused LASSO



- Enforcing temporal coherence leads to adding a penalty term:

$$\widehat{\beta}_\lambda \in \underset{\beta\in\mathbb{R}^d}{\arg\min} \left\{ \|\mathbf{Y} - \mathbf{X}\beta\|^2 + \lambda\|\beta\|_1 + \mu \sum_{j=2}^{d} |\beta_j - \beta_{j-1}| \right\}$$

Sparsity and linear models

Ridge regression

Penalized optimization
Other penalties?

- Until now: hypothesis class with linear functions $h \in \mathcal{H}$ and variations on sparsity-inducing penalties

$$\text{Criterion}(h) = \text{Training error}(h) + \lambda\ \text{Penalty}(h)$$

- This idea goes back to the 60s (Ivanov, John, Lavrent'ev, Tikhonov) where the penalty operated as a regularizer of solutions for ill-posed problems.

Ill-posed problem in statistics
High dimensional least square regression

- Assume d larger than n

- Then when solving the least square optimization problem, we observe that we have less equations than variables: this is the case of an *underdetermined* linear system.

- Another way to put this is to observe that $\mathbf{X}^T\mathbf{X}$ is not full rank, hence it is not invertible and there is an infinity of solutions.

The oldest regularizer in statistics
Ridge regression

- The Ridge estimator is the solution of the following penalized optimization problem: for any $\lambda > 0$,

$$\widehat{\beta}_\lambda \in \operatorname*{arg\,min}_{\beta \in \mathbb{R}^d} \left\{ \|\mathbf{Y} - \mathbf{X}\beta\|^2 + \lambda \|\beta\|_2^2 \right\}$$

Derivation of ridge regression estimator

- We denote the objective function:

$$F(\beta) = (\mathbf{Y} - \mathbf{X}\beta)^T (\mathbf{Y} - \mathbf{X}\beta) + \lambda\beta^T\beta$$

- Thanks to convexity and differentiability of F, we obtain the solution by solving

$$\nabla F(\beta) = 2\mathbf{X}^T(\mathbf{X}\beta - \mathbf{Y}) + 2\lambda\beta = 0$$

- Solution:

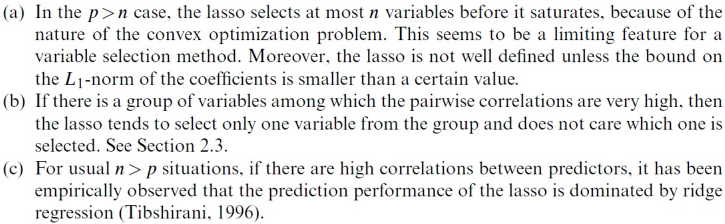
$$\widehat{\beta}_\lambda = \left(\mathbf{X}^T\mathbf{X} + \lambda I_d\right)^{-1} \mathbf{X}^T\mathbf{Y}$$

  because $\mathbf{X}^T\mathbf{X} + \lambda I_d$ always invertible.

- Computation still painful for d very large...

Elastic Net
The best of LASSO and Ridge?

- Rationale (from [Zou and Hastie, 2005])



(a) In the $p > n$ case, the lasso selects at most n variables before it saturates, because of the nature of the convex optimization problem. This seems to be a limiting feature for a variable selection method. Moreover, the lasso is not well defined unless the bound on the L_1-norm of the coefficients is smaller than a certain value.

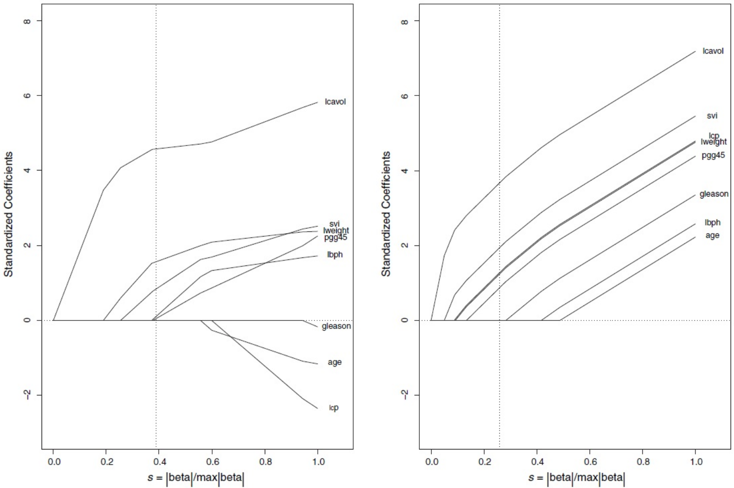
(b) If there is a group of variables among which the pairwise correlations are very high, then the lasso tends to select only one variable from the group and does not care which one is selected. See Section 2.3.

(c) For usual $n > p$ situations, if there are high correlations between predictors, it has been empirically observed that the prediction performance of the lasso is dominated by ridge regression (Tibshirani, 1996).

- Combination of ℓ_1 and ℓ_2 penalties

$$\widehat{\beta}_\lambda \in \arg\min_{\beta\in\mathbb{R}^d} \left\{ \|\mathbf{Y} - \mathbf{X}\beta\|^2 + \lambda\|\beta\|_1 + \mu\|\beta\|_2^2 \right\}$$

LASSO vs. Elastic Net
Comparison of regularization paths



(a)

(b)

Tuning the hyperparameters

- How do we select the parameters λ and μ? These are called hyperparameters or smoothing parameters or regularization parameters.
- This is a universal problem in monitoring the overfitting effect of Machine Learning methods.
- The best practice is a procedure called cross-validation.

Cross-validation

Realistic setup for learning and assessment

- Generic prediction problem with supervised learning: ℓ is a positive loss, f a decision rule

$$L(f) = \mathbb{E}\big(\ell(Y, f(X))\big)$$

- Available data: $D_n = \{(X_i, Y_i) \ : \ i \in \{1, \dots, n\}\}$
- Questions 1: estimate a decision rule $\widehat{f}$
- Question 2: assess performance and estimate $L(\widehat{f})$
- Key ingredients: data splitting and randomization

Strategy 1 for error estimation - Holdout

- Set $D_I = \{(X_i, Y_i) \ : \ i \in I\} \subset D_n$ with index set $I \subset \{1, \ldots, n\}$
- Set an algorithm $\mathcal{A} : D \mapsto \widehat{f}_D$
- Draw $I \subset \{1, \ldots, n\}$
- Estimate of the decision rule $\mathcal{A}(D_I) = \widehat{f}_I$
- Holdout estimate of performance

$$\widehat{L}^H_{n-|I|}(\widehat{f}_I) = \frac{1}{n - |I|} \sum_{i \notin I} \ell(Y_i, \widehat{f}_I(X_i))$$

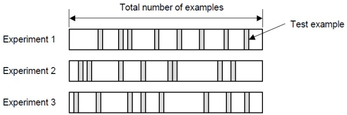
Strategy 2 for error estimation - Bootstrap

- Draw B subsets of indexes $I_1, \ldots, I_B$ of $\{1, \ldots, n\}$ such that $|I_b| = n - m$
- Holdout estimates of the decision rule $\mathcal{A}(D_{I_b}) = \widehat{f}^{\,b}$
- Bootstrap estimate of performance

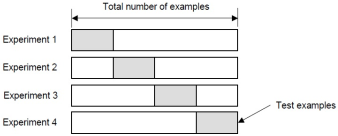
$$\widehat{L}_m^B(\mathcal{A}, D_n) = \frac{1}{B} \sum_{b=1}^{B} \widehat{L}_m^H(\widehat{f}^{\,b})$$

Variants

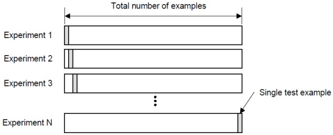
- Random subsampling



- *V*-fold cross-validation



- Leave-one-out
  (and Leave-*p*-out)



Next chapter

- Specific problems: classification, scoring, . . .
- ML algorithms for supervised learning
- Unsupervised learning and dimension reduction